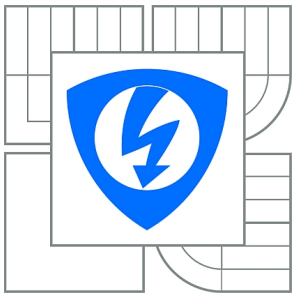
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

GENEROVÁNÍ PÁSMOVĚ OMEZENÝCH ČÍSLICOVÝCH ZVUKOVÝCH SIGNÁLŮ V REÁLNÉM ČASE

REAL-TIME GENERATION OF BAND-LIMITED DIGITAL AUDIO SIGNALS

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

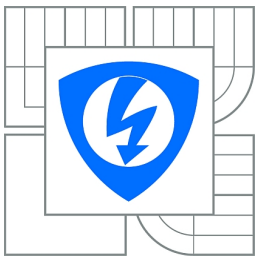
AUTOR PRÁCE — Bc. PETR MAULE

AUTHOR

VEDOUCÍ PRÁCE — Ing. JIŘÍ SCHIMMEL, Ph.D.

SUPERVISOR

BRNO 2010



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Petr Maule	***ID:***	78446
Ročník:	2	***Akademický rok:***	2009/2010

NÁZEV TÉMATU:

Generování pásmově omezených číslicových zvukových signálů v reálném čase

POKYNY PRO VYPRACOVÁNÍ:

Prostudujte metody generování pásmově omezených číslicových zvukových signálů pro hudební syntezátory, které nepoužívají převzorkování, např metodu harmonické syntézy, DSF, BLIT, tabulkové metody a další. Tyto metody porovnejte z hlediska výpočetní náročnosti a možnosti plynulých změn parametrů signálu v reálném čase, např. puslní šířkové modulace. Tyto metody naprogramujte jako jednoduché funkce v prostředí Matlab. Zvolenou metodu impementujte v jazyce C++ jako plug-in modul technologie VST.

DOPORUČENÁ LITERATURA:

[1] Stilson, T., S., Efficieantly-Variable Non-Oversampled Algorithms in Virtual-Analog Music Synthesis - a Root-Locus Perspective. Ph.D. Thesis, Stanford University, 2006.
[2] Moorer, J., A., "The synthesis of complex audio spectra by means of discrete summation formulae". Journal of the Audio Engineering Society, vol. 24, pp. 717–727, Dec. 1975.
[3] Lane, J., Hoory, D., Martinez, E., Wang, P., "Modeling analog synthesis with DSPs". Computer Music Journal, vol. 21, pp. 23–41, 1997.
[4] Valimaki, V., Huovilainen, A., "Oscillator and filter algorithms for virtual analog synthesis". Computer Music Journal, pp. 21–33, 2006.

Termín zadání:	29.1.2010	***Termín odevzdání:***	26.5.2010

Vedoucí práce: Ing. Jiří Schimmel, Ph.D.

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

ANOTACE

Diplomová se zabývá problematikou generování číslicových zvukových signálů s pásmově omezeným kmitočtovým spektrem, tj. bez aliasingového zkreslení. V teoretické části práce jsou popsány jednotlivé metody generování pásmově omezených obdélníkových, trojúhelníkových a pilových průběhů v reálném čase. Popsané metody jsou naprogramovány v prostředí Matlab a porovnány z hlediska změn parametrů v reálném čase, jako jsou například změna střídy obdélníkového průběhu nebo plynulá změna kmitočtu.

Hlavní část práce je věnována implementaci metod postupných integrací pásmově omezeného sledu impulzů a metodě derivací po částech parabolického průběhu v jazyce C++. Implementované metody byly integrovány do plug-in modulu technologie VST generující zvukový signál v reálném čase. Implementované metody jsou porovnány z hlediska výpočetní složitosti a zkreslení generovaného signálu.

KLÍČOVÁ SLOVA

Metoda BLIT, metoda DPW, pásmově omezený, metoda DSF, aliasing, technologie VST, generování číslicových zvukových signálů, zpracování signálu v reálném čase.

ABSTRACT

Master's thesis deals with the generation of digital audio signals with band-limited frequency spectrum, i.e. without the aliasing distortion. Various methods of generating band-limited rectangular, triangular, and sawtooth waveforms are described in the theoretical part. The described methods are programmed in the Matlab programming environment and compared in terms of real-time parameter changes, such as duty cycle change of rectangular waveform or continuous change of frequency.

The main part of the thesis describes implementation of methods of successive integration of band-limited impulse train and method of differentiated parabolic waveforms in C++ language. The implemented methods were integrated into a plug-in of VST technology that generates an audio signal in real time. The implemented methods are compared in terms of computational complexity and distortion of the generated signal.

KEYWORDS

BLIT method, DPW method, Band-limited, DSF method, Aliasing, VST technology, digital audio signal generation, Real-time signal processing.

MAULE, P. *Generování pásmově omezených číslicových zvukových signálů v reálném čase*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2010. 57 s., 1 s. přílohy. Vedoucí diplomové práce Ing. Jiří Schimmel, Ph.D.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma Generování pásmově omezených číslicových zvukových signálů v reálném čase jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením tohoto projetu jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení §152 trestního zákona č. 140/1961 Sb.

V Brně dne…….. …….
podpis autora

PODĚKOVÁNÍ

Děkuji vedoucímu diplomové práce Ing. Jiřímu Schimmelovi, Ph.D. za velmi užitečnou metodickou pomoc, poskytnuté odborné materiály a cenné rady při zpracování diplomové práce.

V Brně dne…….. …….
podpis autora

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM ZKRATEK, VELIČIN A SYMBOLŮ

ASIO	(Audio Stream Input/Output) ovladač zvukové karty s malým dopravním zpožděním
BLIT	(Bandlimited Impulse Train) pásmově omezený sled impulzů
BP-BLIT	(Bipolar Bandlimited Impulse Train) bipolární pásmově omezený sled impulzů
$CIT(\tau)$	(Continuous-time Impulse Train) periodický sled impulzů se spojitým časem
CPU	(Central Processing Unit) procesor
d	střída obdélníkového signálu
DDR	Double-Data Rate
DLL	(Dynamic-Link Library) dynamicky připojovaná knihovna
DPW	(Differentiated Parabolic Waves) derivace parabolických průběhů
DSF	(Discrete Summation Formulas) diskrétní součtový vzorec
f	kmitočet [Hz]
f_0	základní generovaný kmitočet [Hz]
FIR	(Finite Impulse Response) konečná impulzová odezva
f_m	mezní kmitočet [Hz]
f_{vz}	vzorkovací kmitočet
$g(f,d)$	funkce kmitočtu a střídy
GUI	(Graphic User Interface) grafické uživatelské rozhranní
$H(z)$	přenosová funkce systému s diskrétním časem
$h[n]$	impulzová charakteristika
$H_D(z)$	přenosová funkce dolní propusti
h_{vz}	impulzová odezva anti-aliasingového filtru
IIR	(Infinite Impulse Response) nekonečná impulzová odezva
k_0	délka impulzu ve vzorcích
M	počet harmonických složek v oboustranném spektru
N	délka posloupnosti vzorků nebo délka tabulky
n	index vzorku
Num	požadovaný počet harmonických složek
OS	(Oprating System) operační systém
P	perioda ve vzorcích
PWM	(Pulse Width Modulation) impulzní šířková modulace
RAM	(Random Access Memory) paměť s náhodným přístupem
$rect[n]$	obdélníkový signál s diskrétním časem
$rect_{CTS}(t)$	obdélníkový signál se spojitým časem
$S(k)$	spektrum obecného diskrétního signálu
$s(t)$	signál se spojitým časem
$s[n]$	signál s diskrétním časem
$saw[n]$	pilový signál s diskrétním časem
$saw_{CTS}(t)$	pilový signál se spojitým časem
$s_v(t)$	vzorkovaný signál
$S_v(\omega)$	spektrální funkce vzorkovaného signálu
$s_\delta(t)$	periodický sled jednotkových impulzů
t	čas [s]
THD	(Total Harmonic Distortion) celkové harmonické zkreslení
$tri[n]$	trojúhelníkový signál s diskrétním časem

$tri_{CTS}(t)$	trojúhelníkový signál se spojitým časem
T_{vz}	vzorkovací perioda
VST	(Virtual Studio Technology) technologie plug-in modulů pro generování a zpracování zvukových signálů v reálném čase na osobním počítači
VST SDK	(Virtual Studio Technology Software Development Kit) vývojové prostředky technologie VST
$w(t)$	váhovací okno
WDM	(Windows Driver Model) standardní ovladače hardware pro operační systém Windows
$x_f(t)$	pásmově omezený signál
$y[n]$	výstupní diskrétní signál generující sled impulzů
ω	úhlový kmitočet [rad · s^{-1}]
ω_m	mezní úhlový kmitočet [rad · s^{-1}]

ÚVOD

Uměle generované zvukové signály jsou produktem syntezátorů. V minulosti byly velmi populární analogové syntezátory. V současnosti, s velkým rozvojem digitální techniky, se do popředí zájmu stále častěji dostávají digitální syntezátory. Zvuk, který je generovaný syntezátory, je využíván při nejrůznějších příležitostech nejen v počítačových hrách a ve zvukových efektech doprovázejících činnosti prováděné na počítači, ale i pro generování zvuků doprovázejících profesionální hudební nahrávky ba dokonce k syntéze celých hudebních nahrávek některých hudebních žánrů. První syntezátory založené na analogových principech se začaly objevovat v šedesátých a sedmdesátých letech minulého století. V osmdesátých letech se začaly využívat digitální syntezátory. V dnešní době jsou číslicové syntezátory i takové, kde procesor digitálního syntezátoru neprovádí vlastní softwarovou syntézu, ale simuluje činnost analogového syntezátoru. Jednou z metod využívaných v číslicových syntezátorech zvuku je subtraktivní syntéza, která pro svoji činnost vyžaduje signály bohaté na spektrální složky, jako jsou klasické analogové signály s trojúhelníkovým, pilovým a obdélníkovým průběhem. Jejich syntézou se zabývá následující práce.

Základní signály s nekonečným spektrem harmonických složek musejí být při vzorkování pásmově omezeny s mezním kmitočtem menším než je polovina vzorkovacího kmitočtu. Při jejich číslicové syntéze platí stejná podmínka. Jednou z možností, jak tyto signály syntetizovat, je využití metod založených na principu převzorkování, kdy je využíván vysoký vzorkovací kmitočet. To však klade na číslicový systém značné výpočetní nároky, což samozřejmě vede k vyšším nákladům na systém. Ke snížení nákladů na tvorbu číslicových systémů byly vyvinuty metody, které princip převzorkování nevyužívají na úkor snížení kvality generovaného signálu.

Práce je koncipována do pěti kapitol. První kapitola se zabývá možnostmi generování trojúhelníkového, pilového a obdélníkového signálu v „analogových“ dobách a také zde bude objasněn vznik aliasingu při vzorkování těchto signálů.

Následující kapitola popisuje některé metody pro generování pásmově omezených signálů, které nevyužívají převzorkování, a ukazuje možnosti, jak těmito metodami generovat pilový, obdélníkový a trojúhelníkový průběh.

Ve třetí kapitole je provedeno porovnání vybraných metod z hlediska změn parametrů v reálném čase, např. změny kmitočtu nebo střídy obdélníkového průběhu.

Porovnání je provedeno na základě simulací vytvořených v programovacím prostředí Matlab.

Podstatou posledních dvou kapitol je implementace vybraných metod, které byly popsány ve třetí kapitole. Metody jsou naprogramovány v jazyce C++ a z nich vytvořen plug-in modul technologie VST, který slouží ke generaci pilového, obdélníkového a trojúhelníkové průběhu. V závěrečné kapitole jsou porovnány reálné parametry implementovaných metod.

1 ZÁKLADNÍ SIGNÁLY

1.1 Analogové generování pásmově neomezených signálů

Klasické signály analogových syntezátorů, které se musely generovat, byly pilové, obdélníkové a trojúhelníkové. Většinou nebylo nutné generovat sinusové průběhy, protože analogové syntezátory využívaly subtraktivní syntézu, která ve své podstatě vyžaduje signály bohaté na spektrální složky, ale sinusový průběh obsahuje pouze jednu harmonickou složku.

Pilový průběh byl obvykle generován pomocí integrátoru s nastavitelným ziskem, tím je regulován sklon pilového průběhu. Pokud tento integrátor dosáhne prahové hodnoty, dochází k integraci od počátku.

Obdélníkový průběh byl obvykle vytvářen pomocí pilového průběhu s určitou stejnosměrnou složkou, který byl přiváděn na komparátor. Střída obdélníkového průběhu byla regulována podle prahové hodnoty komparátoru. Další možností, jak vytvořit obdélníkový průběh, bylo odečíst od sebe dva pilové průběhy, kde jeden měl určité zpoždění vůči druhému. V analogové technice se tato možnost ale příliš nevyužívala z důvodu náročnosti vytvoření zpoždění. Naopak je často využívána k syntéze signálů v číslicové technice.

Trojúhelníkový průběh byl realizován mnoha způsoby, jedním z nich je například integrace obdélníkového průběhu [1].

1.2 Aliasing

Máme-li spojitý signál $s(t)$ a periodický sled jednotkových impulzů $s_\delta(t)$, dostaneme ideální vzorkování vynásobením těchto dvou signálů. Vzorkovaný signál je tedy dán vztahem [2]

$$s_v(t) = s(t)s_\delta(t) = s(t)\sum_{n=-\infty}^{\infty}\delta(t - nT_{vz}), \tag{1.1}$$

kde T_{vz} je perioda vzorkování. Spektrální funkce součinu signálů je dána konvolucí jejich spektrálních funkcí. Pro spektrální funkci signálu $s_v(t)$ tedy platí [2]

$$S_v(\varpi) = f_{vz}\sum_{k=-\infty}^{\infty}S(\varpi - k\varpi_{vz}). \tag{1.2}$$

Vidíme, že spektrum vzorkovaného signálu je zperiodizované podle vzorkovacího kmitočtu f_{vz} a musíme tedy dodržet vzorkovací teorém $f_{vz} > 2f_m$, protože jinak nastane aliasing [2]. U pilového obdélníkového a trojúhelníkového signálu nemůžeme dodržet tento vzorkovací teorém, protože mají nekonečné spektrum, jak vyplývá z uvedených Fourierových řad těchto signálů [3]:

- Obdélníkový průběh se střídou 50 %

$$S(k) = \frac{4}{\pi}\sum_{k=1}^{\infty}\frac{1}{k}\sin(2\pi fkt) \qquad \text{pro k = 1,3,5,…} \tag{1.3}$$

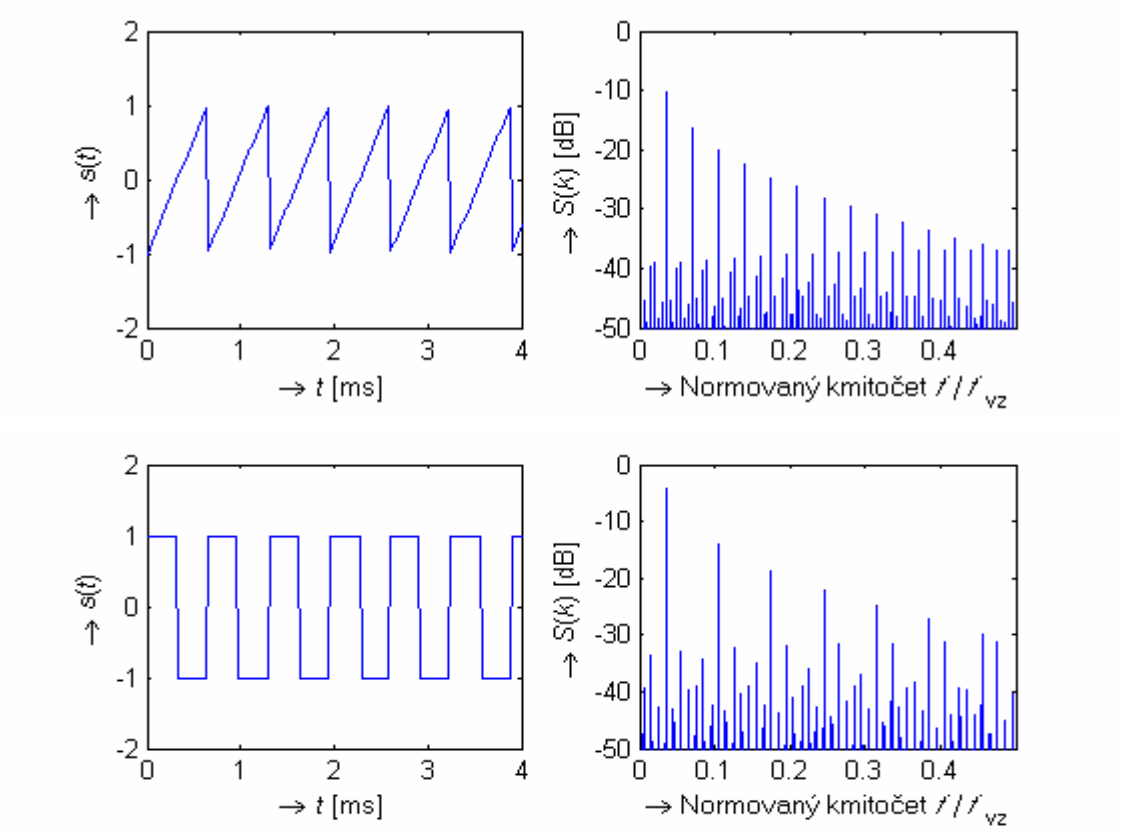
- Pilový průběh

$$S(k) = -\frac{2}{\pi}\sum_{k=1}^{\infty}\frac{1}{k}\sin(2\pi fkt) \qquad \text{pro k = 1,2,3,…} \tag{1.4}$$

- Trojúhelníkový průběh

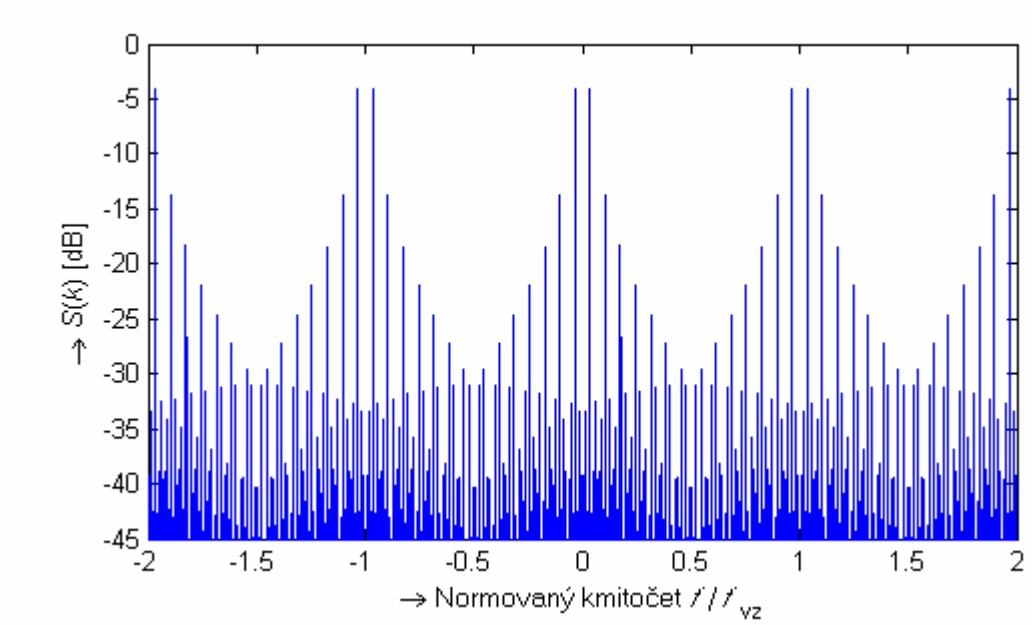
$$S(k) = \frac{8}{\pi^2}\sum_{k=1}^{\infty}\frac{1}{k^2}(-1)^{\frac{k-1}{2}}\sin(2\pi fkt) \qquad \text{pro k = 1,3,5,…} \tag{1.5}$$

Uvedené signály musíme pásmově omezit s mezním kmitočtem $f_{vz}/2$ v souladu se vzorkovacím teorémem, protože by nastal aliasing, jak znázorňuje obr. 1.1.



Obr. 1.1: Spektra jednoduchého pilového a obdélníkového průběhu s masivním aliasingem, který se zobrazí mezi správné harmonické složky.

Na obr. 1.1 je znázorněn pásmově neomezený jednoduchý obdélníkový a pilový průběh se vzorkovacím kmitočtem 44,1 kHz s masivním aliasingem. Vzniklý aliasing je velmi slyšitelný ve zvukovém signálu. Nejmenší aliasing je patrný na nízkých kmitočtech v porovnání se vzorkovacím kmitočtem, naopak největší aliasing vzniká na kmitočtech blízkých polovině vzorkovacího kmitočtu. Na obr. 1.2 je znázorněno zperiodizované spektrum jednoduchého obdélníkového průběhu s masivním aliasingem.



Obr. 1.2: Zperiodizované spektrum jednoduchého obdélníkového průběhu s masivním aliasingem.

2 METODY SYNTÉZY PÁSMOVĚ OMEZENÝCH SIGNÁLŮ

Metody pro syntézu pásmově omezených signálů popsané v této kapitole lze rozdělit do tří základních skupin [4]:

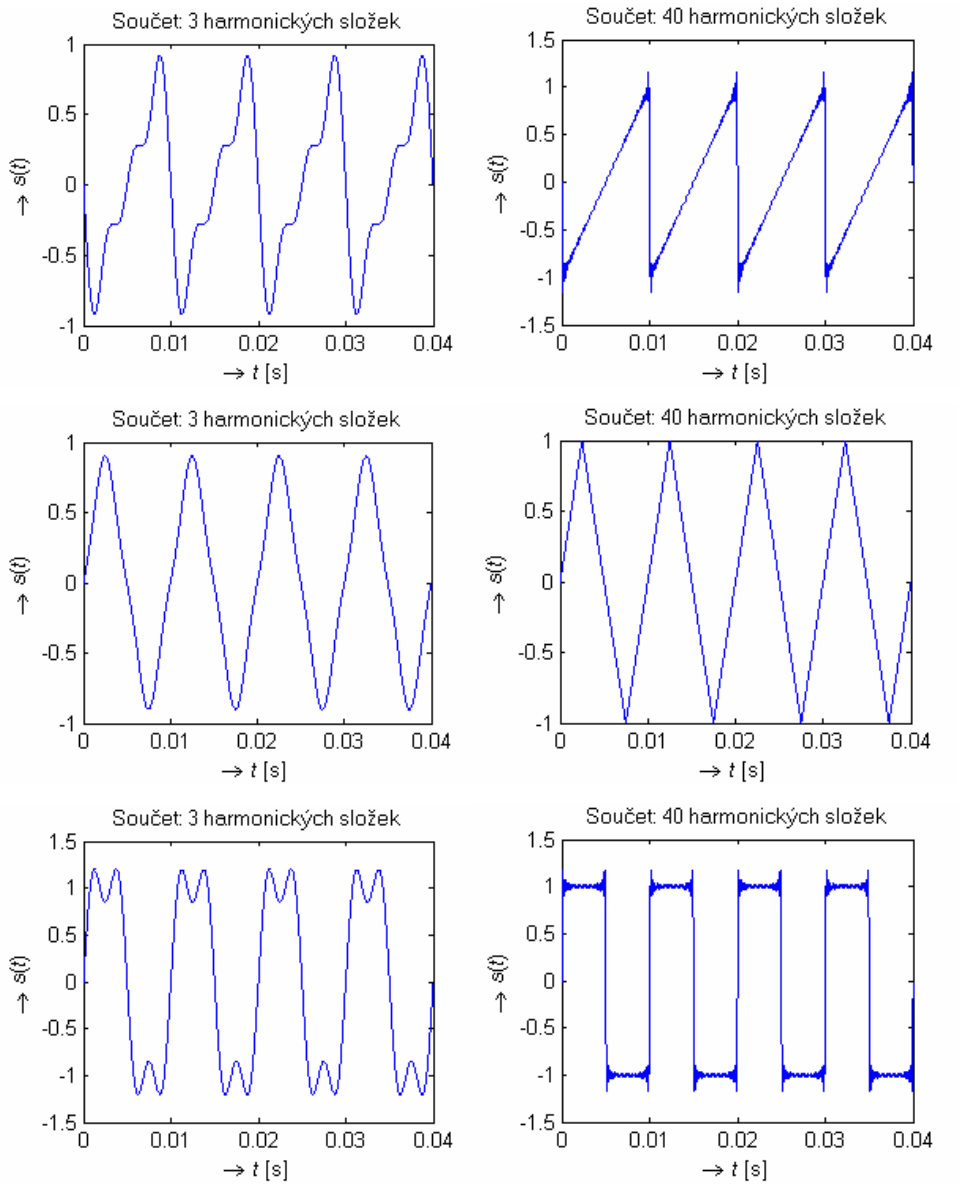
- První skupinou jsou metody, které generují signály s pevným počtem harmonických složek. Patří sem metody jako aditivní syntéza a metoda postupných integrací pásmově omezeného sledu impulzů, kde je sled impulzů generován metodou s pevným počtem harmonických složek [4]. Tyto metody zcela zabraňují vzniku aliasingového zkreslení.
- Druhou skupinou jsou metody, které udržují určitou požadovanou hodnotu aliasingového zkreslení na základě filtrace signálu. Do této skupiny spadá například metoda postupných integrací, kde pásmově omezený sled impulzů je syntetizován na základě okénkovaných funkcí sinc (BLIT-SWS – Sum of Windowed Sinc) [4].
- Poslední skupinou jsou metody, které potlačují aliasingové zkreslení na základě změny poklesu velikostí spektrálních složek generovaných signálů. Tato skupina obsahuje například metodu derivací po částech parabolických průběhů (DPW - Differentiated Parabolic Waves) [4].

2.1 Aditivní syntéza

Aditivní syntéza je nejednodušší cestou, jak generovat základní průběhy s pásmově omezeným spektrem. Základní myšlenkou této syntézy je, že každý složitější periodický signál se dá rozložit na součet sinusových a kosinusových průběhů. Daný koncept se nazývá Fourierova transformace [2]. Na obr. 2.1 jsou zobrazeny vygenerované průběhy pilového, obdélníkového a trojúhelníkového signálu metodou aditivní syntézy.

Tato metoda patří mezi nejdražší a výpočetně nejnáročnější, protože je potřeba velké množství oscilátorů ke generování požadovaného průběhu. Například pro generování pilového průběhu s kmitočtem 30 Hz je potřeba 735 harmonických složek, jestliže využijeme celé pásmo - tedy do poloviny vzorkovacího kmitočtu, zatímco při generaci pilového průběhu o kmitočtu 3000 Hz potřebujeme pouze 7 harmonických složek. (Vzorkovací kmitočet má obvykle hodnotu 44100 Hz při audio kvalitě.) Z čehož vyplývá i použití pouze v hybridních systémech, kde je aditivní syntéza použita ke

generaci vysokých kmitočtů. Zde už je počet potřebných harmonických složek velmi malý. Nejčastěji je aditivní syntéza využívána s 32 až 64 harmonickými složkami [3], ve starších hudebních syntezátorech s 10 harmonickými složkami.



Obr. 2.1: Průběhy pilového, trojúhelníkového a obdélníkového signálu vygenerované aditivní syntézou pro první tři a čtyřicet harmonických složek.

2.2 Tabulková metoda

Jedná se o jednu z nejstarších technik syntézy používané v digitální hudební technice. V této metodě jsou v tabulce uloženy pouze vzorky jedné periody. Používaná délka tabulky (počet vzorků jedné periody) je od 128 do 2048 vzorků. Výstupní signál se vytváří periodickým vyčítáním tabulky délky N se základním kmitočtem $f = f_{vz}/N$. Vyšší

kmitočty, například dvojnásobný, se vytváří čtením každého druhého vzorku z tabulky na výstup $2f = 2f_{vz}/N$. Jiný kmitočet můžeme získat pomocí tzv. fázového přírůstku *inc* mezi následujícími dvěma vzorky:

$$inc = N\frac{f_0}{f_{vz}}, \tag{2.1}$$

kde f_0 je základním kmitočtem generovaného signálu. To ovšem funguje, obsahuje-li tabulka dostatečné množství vzorků. V případě, že fázový přírůstek vychází neceločíselný, je několik možností. Buď je zaokrouhlen na nejbližší celé číslo nebo je jeho hodnota interpolována nejčastěji pomocí metody lineární interpolace, případně použitím pásmově omezené interpolace [5].

U interpolované tabulky není zaručen pásmově omezený výstupní signál, když je přírůstek větší než jeden vzorek. Proto pro tabulky, které generují jiné než sinusové signály, musí být ve fázovém přírůstku zahrnuta i nejvyšší harmonická složka signálu v tabulce

$$inc_{MAX} = N\frac{f_0}{nf_{vz}}, \tag{2.2}$$

kde *n* je číslo nejvyšší harmonické složky. Interpolovaná tabulková metoda je v zásadě shodná s aditivní syntézou, která vyžaduje digitální sinusový oscilátor pro každou harmonickou [1].

2.3 Syntéza metodou diskrétního součtového vzorce

Diskrétní součtový vzorec (DSF-Discrete Summation Formulas) je používán pro syntézu pásmově omezených periodických signálů. Tato metoda může být použita pro syntézu kmitočtové modulace a mnoho dalších periodických průběhů. Základní vzorec DSF je [6]

$$\sum_{k=0}^{Num} a^k \cos(\theta + k\beta) = \frac{\sin\theta - a\sin(\theta-\beta) - a^{Num+1}\left[\sin\{\theta + (Num+1)\beta\} - a\sin\right](\theta + Num\beta)}{1 + a^2 - 2a\cos\beta}, \tag{2.3}$$

kde proměnná *Num* udává počet harmonických složek, čímž kontroluje pásmové omezení a musí být nastavena tak, aby nejvyšší harmonická složka byla menší než $f_{vz}/2$, a tím nedošlo k porušení podmínky vzorkování. Nastavení parametrů θ a β je například pro generaci pásmově omezeného sledu impulzů θ na 0 a β na $2\pi ft$ [1]. Parametr *a* udává

pokles amplitudy jednotlivých harmonických. To znamená, že jestliže $a = 0{,}5$, pak pokles mezi následujícími dvěma harmonickými bude o 6 dB.

Metodu DSF nelze použít k přímé syntéze trojúhelníkového, pilového ani obdélníkového průběhu, protože kupříkladu obdélníkový průběh má pokles amplitudy harmonických složek $1/f$ a trojúhelníkový průběh dokonce $1/f^2$ a jak bylo uvedeno, DSF má pokles pouze a^k. Z toho vyplývá. že metodu DSF nelze využít pro generování přesných průběhů trojúhelníkového ani obdélníkového tvaru, lze ji tedy použít pouze ke generaci nepřesných signálů pro řídící účely.Více o dané metodě se lze dočíst v [6].

2.4 Metoda derivace parabolického průběhu

Metoda derivace po částech parabolického průběhu (DPW - Differentiated Parabolic Waves) je začleněna do skupiny metod, které redukují aliasingové zkreslení na základě změny poklesu spektrálních složek signálu. Motivací k použití parabolického průběhu je to, že velikosti harmonických složek parabolického průběhu klesají s rychlostí 12 dB na oktávu místo poklesu 6 dB na oktávu u pilového průběhu [7]. Derivace obnovuje požadovaný pokles velikostí harmonických složek, ale v nízkých a středních kmitočtech spektra mají aliasingové složky nižší úroveň než u jednoduchého pilového průběhu.

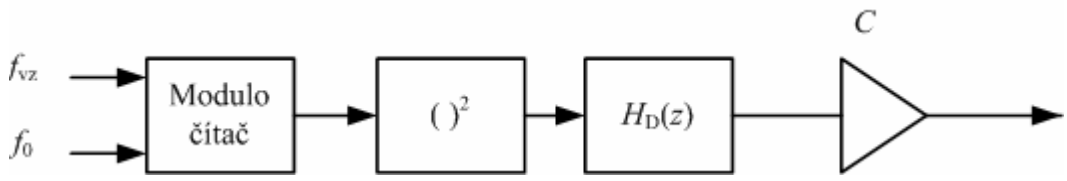
2.4.1 Generace pilového průběhu derivací parabolického průběhu

Vesa Välimäki si jako první uvědomil, že digitální signál, který se velmi podobá pilovému průběhu ovšem s výrazně nižším aliasingem, lze získat z derivace po částech parabolického průběhu DPW [8]. Základní způsob tohoto principu generování je zobrazen na obr. 2.1. Nejdříve je vytvořen jednoduchý pilový průběh pomocí bipolárního modulo čítače. Jednoduchý pilový průběh s amplitudou od -1 do 1 a s diskrétním časem je vytvořen podle rovnice [7]

$$s[n] = 2\left(n\frac{f_0}{f_{vz}} \bmod 1 \right) - 1, \tag{2.4}$$

kde f_0 je kmitočet generovaného průběhu a n je index vzorku v diskrétním čase.

Dále jsou hodnoty průběhu umocněny na druhou, čímž je vytvořen po částech parabolický průběh funkce. Poté je signál derivován, což se děje filtrováním pomocí číslicového filtru prvního řádu s přenosovou funkcí $H_D(z) = 1 - z^{-1}$.



Obr. 2.2: Základní princip generování pilového průběhu pomocí DPW.

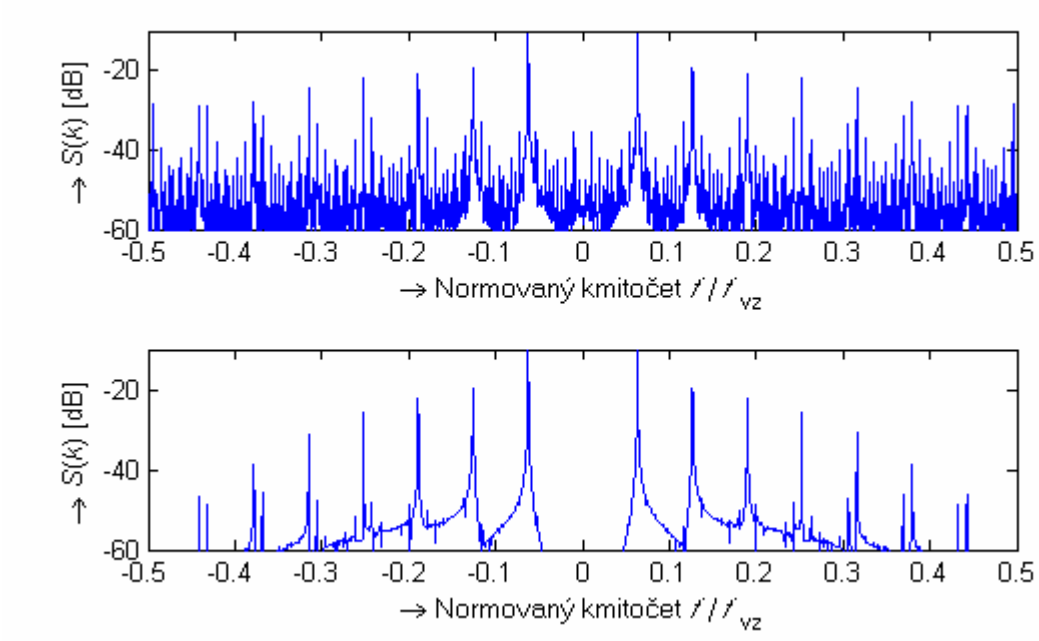
Použijeme-li pouze tento jednoduchý derivátor, nezajistíme tím pásmově omezený signál a v signále bude stále značné aliasingové zkreslení, proto je nutné tento filtr nahradit tzv. average filtrem, který má přenosovou funkci [8]

$$H_D(z) = \frac{z^2 - 1}{2z^2}. \tag{2.5}$$

Nakonec jsou hodnoty průběhu násobeny konstantou

$$c = \frac{f_{vz}}{4 f_0 (1 - f_0 / f_{vz})}, \tag{2.6}$$

tak, aby byl výsledný průběh v rozmezí od -1 do 1. Na obr. 2.3 je znázorněno porovnání spekter jednoduchého pilového průběhu s průběhem, který vznikl generací pomocí metody DPW. Tento průběh má značně potlačené aliasingové zkreslení. Více o této metodě lze nalézt v [4], [8].

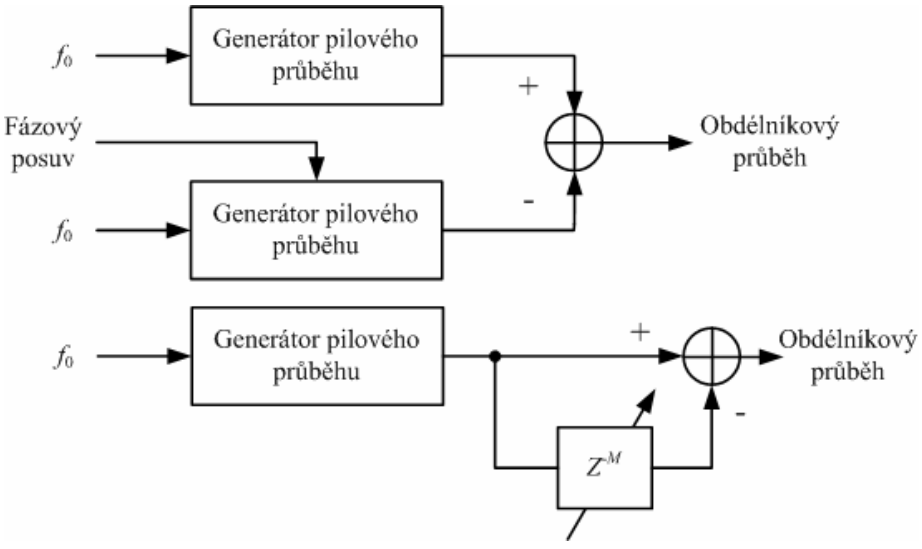


Obr. 2.3: Spektrum jednoduchého pilového průběhu o základním kmitočtu 2783,4 Hz a průběhu generovaného pomocí DPW.

2.4.2 Generování obdélníkového průběhu pomocí pilového průběhu

Obdélníkový průběh lze získat pomocí dvou způsobů, jak je naznačeno na obr. 2.4. První metoda je za použití dvou generátorů pilového průběhu, kde jeden generátor má nastavitelný fázový posuv, poté se výstupy obou generátorů sečtou. Jako generátory pilového průběhu použijeme výše popsanou generaci za pomoci DPW. Tato metoda umožňuje plynulou impulzní šířkovou modulaci [8].

Další možnou metodou je použití zpožďovacího článku a pouze jednoho generátoru pilového průběhu. To vyžaduje další proměnlivé paměťové nároky pro zapamatování vzorků. Paměť je závislá na počtu vzorků, o které je třeba signál zpozdit (požadovaná střída výsledného obdélníkového signálu). Navíc při použití této metody lze fázi měnit pouze o celočíselné násobky vzorků, ale ne plynule jako u předchozí metody, čímž ve výsledném signálu vzniká přídavný šum (zipper noise). Výpočetní náročnost zpožďovacího článku a jednoho generátoru pilového průběhu je nakonec téměř stejná jako u předchozí metody s použitím dvou generátorů. Obě uvedené metody mají výsledný obdélníkový průběh s téměř nulovou stejnosměrnou složkou [8].

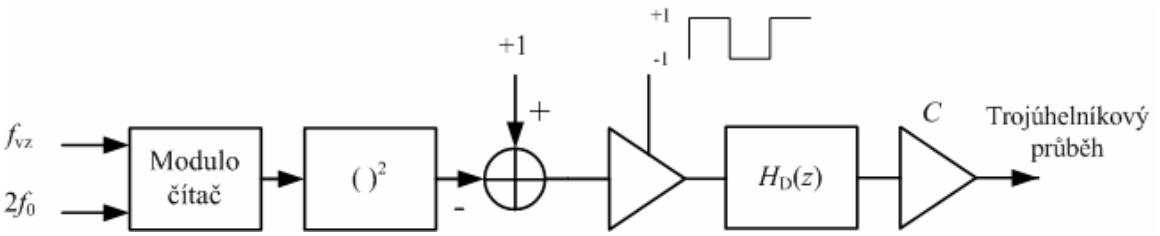


Obr. 2.4: Možnosti generace obdélníkového průběhu.

2.4.3 Generování trojúhelníkového průběhu metodou derivace parabolického průběhu

Číslicový trojúhelníkový průběh nemá tak závažné problémy s aliasingem jako pilový nebo obdélníkový signál, protože spektrální složky trojúhelníkového průběhu klesají o 12 dB na oktávu a ne o 6 dB na oktávu [8]. Proto již generátor jednoduchého

trojúhelníkového průběhu může poskytnout dostatečně kvalitní průběh pro subtraktivní syntézu. Na obr. 2.5 je naznačen princip generování trojúhelníkového signálu pomocí bipolární po částech parabolické křivky. Nejdříve je pomocí modulo čítače vygenerován jednoduchý pilový průběh s kmitočtem dvakrát větším než kmitočet požadovaného trojúhelníkového signálu. Po umocnění jsou hodnoty signálu odečteny od jedné. Signál je dále modulován jednoduchým obdélníkovým průběhem s kmitočtem $2f_0$, abychom dostali požadovaný bipolární po částech parabolický signál. Obdélníkový průběh musí být se signálem ve fázi, což se dá jednoduše realizovat přepínáním z -1 na hodnotu +1 pokaždé, když modulo čítač dosáhne hodnoty +1. Nakonec již stačí jen bipolární, po částech parabolický signál derivovat a jeho hodnoty vynásobit konstantou c (2.6). Výsledkem je trojúhelníkový průběh s výrazně nižším aliasingovým zkreslením.



Obr. 2.5: Princip generování trojúhelníkové průběhu metodou DPW.

2.5 Postupné integrace pásmově omezeného sledu impulzů

Získání signálu je možné pomocí lineárních a nelineárních operací. Lineární operace nebudou ve výstupním signálu generovat další spektrální složky, které nejsou obsaženy ve vstupním signálu, tedy pokud bude vstupní signál pásmově omezený, bude i výstupní signál pásmově omezený.

Ukazuje se, že je jednodušší vygenerovat pásmově omezený sled impulzů (BLIT – Bandlimited Impulse Train, metody jeho generace budou uvedeny v následující kapitole) než přímo generovat složitější signál [1]. Pomocí jedné integrace je možné získat pilový i obdélníkový signál a pomocí druhé integrace je možné z obdélníkového průběhu získat trojúhelníkový signál.

2.5.1 Generace pilového průběhu z pásmově omezeného sledu impulzů

Spojitý pilový průběh může být generován podle vztahu [9]

$$saw_{CTS}(t) = \int_0^t CIT(\tau) - C_1 d\tau, \tag{2.7}$$

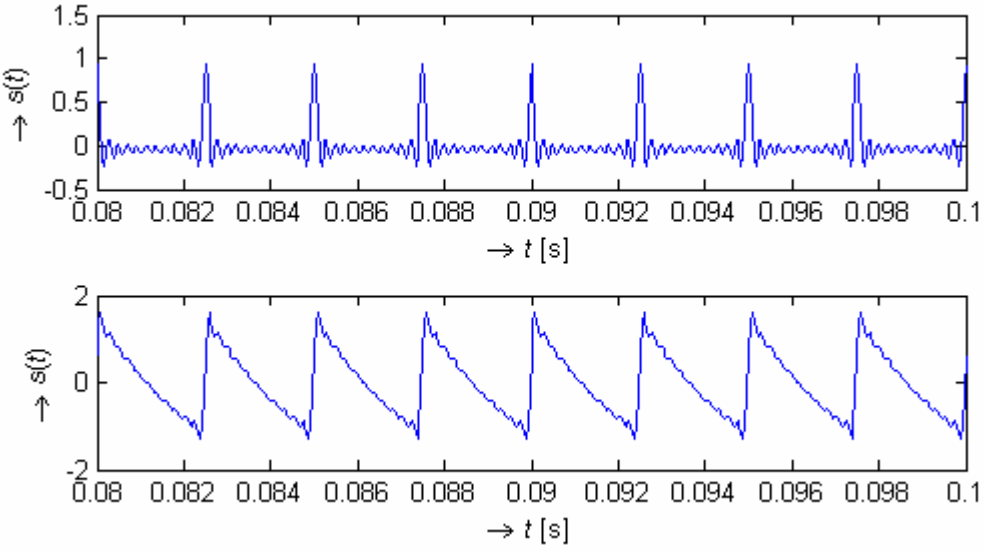
kde $CIT(\tau)$ je periodický sled impulzů se spojitým časem a $C_1 = \int_0^T CIT(\tau) d\tau$, což je stejnosměrná složka periodického sledu impulzů. Vztah (2.7) převedeme do diskrétního času

$$saw[n] = \sum_{k=0}^{N} BLIT[k] - C_2, \tag{2.8}$$

kterému v Z-transformaci odpovídá

$$saw(z) = \frac{z}{z-1}(BLIT(z) - z(C_2)). \tag{2.9}$$

Vztah (2.9) vyjadřuje průchod periodického sledu impulzů číslicovým filtrem prvního řádu typu dolní propust, jak ukazuje obr. 2.6. Stejnosměrná složka C_2 je střední hodnota sledu impulzů, která musí být odečtena, jinak by došlo k saturaci integrátoru. Konstanta C_2 je funkcí kmitočtu. V závislosti na počátečních podmínkách integrátoru bude mít výstupní průběh stejnosměrnou složku [9].



Obr. 2.6: Příklad vygenerovaného pilového průběhu z BLITu s 10 harmonickými složkami.

2.5.2 Generace obdélníkového průběhu z bipolárního sledu impulzů

Spojitý obdélníkový průběh lze vygenerovat podle [9]

$$rect_{CTS}(t) = \int_0^t CIT(\tau) - CIT(\tau - t_0) - C_3 d\tau \,. \tag{2.10}$$

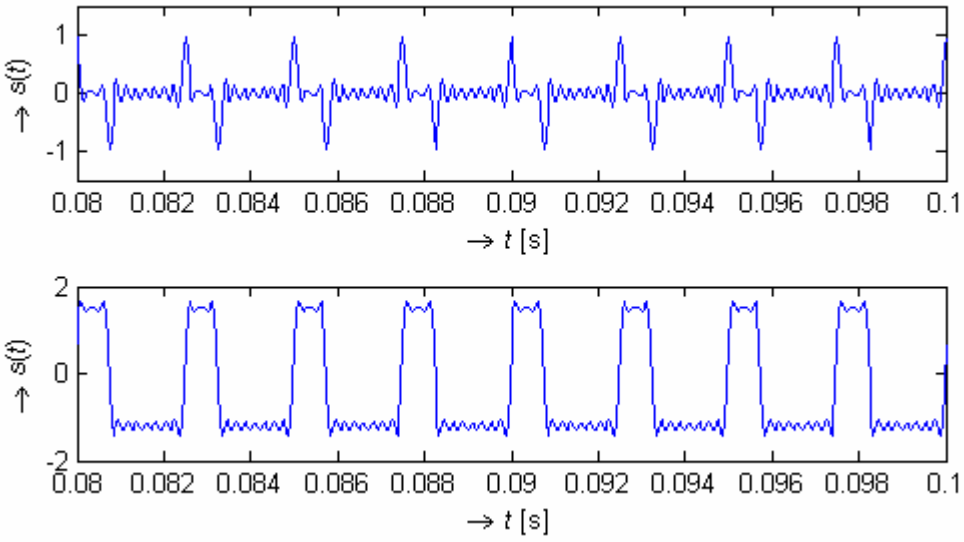
Převedeme-li tento vztah do diskrétního času získáme vztah

$$rect[n] = \sum_{k=0}^{N} BLIT[k] - BLIT[k - k_0] - C_4 = \sum_{k=0}^{N} BP - BLIT_{k_0}[k] - C_4 \,, \tag{2.11}$$

kterému v *Z*-transformaci odpovídá

$$rect(z) = \frac{z}{z-1}\left(BP - BLIT_{k_0}(z) - z(C_4)\right), \tag{2.12}$$

kde BP-BLIT je bipolární sled impulzů. Bipolární sled impulzů má nulovou stejnosměrnou složku. Díky tomu i konstanta C_3 = 0 a následovně C_4 = 0. Střída obdélníku je nastavována pomocí parametru k_0, což je délka pulzu ve vzorcích. Proměnná délka k_0 může být tedy v rozmezí od nuly do délky periody BLIT, tím můžeme získat PWM (impulzní šířková modulace), která má střídu v rozmezí od 0 do 1. Výstupní signál bude mít opět stejnosměrnou složku závisející na počátečních podmínkách integrace. Ukázka získaného obdélníkového průběhu je uvedena na obr. 2.7.



Obr. 2.7: Ukázka obdélníkového signálu získaného z BP-BLT s 10 harmonickými složkami.

2.5.3 Generace trojúhelníkového signálu z obdélníkového průběhu

Spojitý trojúhelníkový signál můžeme získat z obdélníkového signálu dle [9]

$$tri_{CTS}(t) = \int_0^t rect_{CTS}(\tau) - C_5 d\tau\,, \tag{2.13}$$

kde C_5 je stejnosměrná složka obdélníkového průběhu: $C_5 = \int_0^T rect_{CTS}(\tau) d\tau$. Převedeme-li vztah (2.13) do diskrétního času získáme novou rovnici

$$tri[n] = \sum_{k=0}^{N} rect[k] - C_6\,, \tag{2.14}$$

které opět v *Z*-transformaci odpovídá

$$tri(z) = \frac{z}{z-1}\left(rect(z) - z(C_6)\right). \tag{2.15}$$

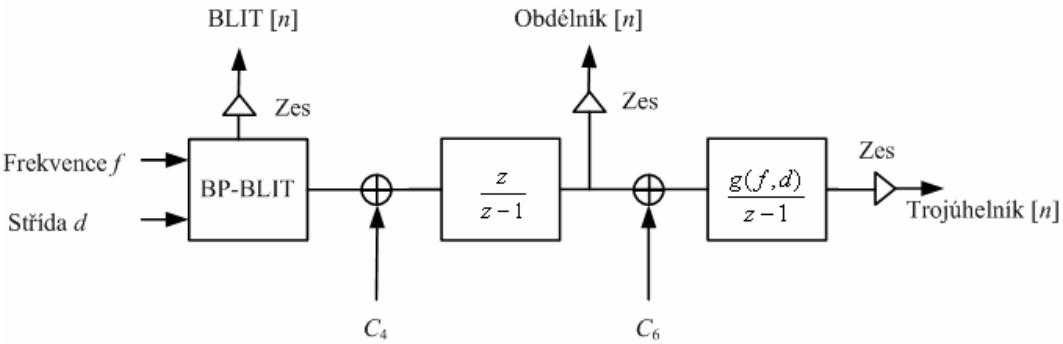
Konstanty C_5 a C_6 jsou funkcí obdélníkového průběhu závislé na jeho střídě a stejnosměrné složce, která vyplývá z počáteční podmínky integrace obdélníkového průběhu $C_6 = k_0 / T + C_7$, kde T je perioda obdélníkového signálu a C_7 je funkce BLIT(0), což je počáteční podmínka první integrace. Princip získání obdélníkového průběhu je dobře zobrazen na obr. 2.8. Chceme-li získat trojúhelníkový průběh stejného rozkmitu jaký má obdélníkový průběh, musíme provést integraci obdélníkového průběhu závislou na kmitočtu a střídě tohoto průběhu

$$tri[n] = \sum_{k=0}^{N} g(f,d)\left(rect[k] - C_6\right), \tag{2.16}$$

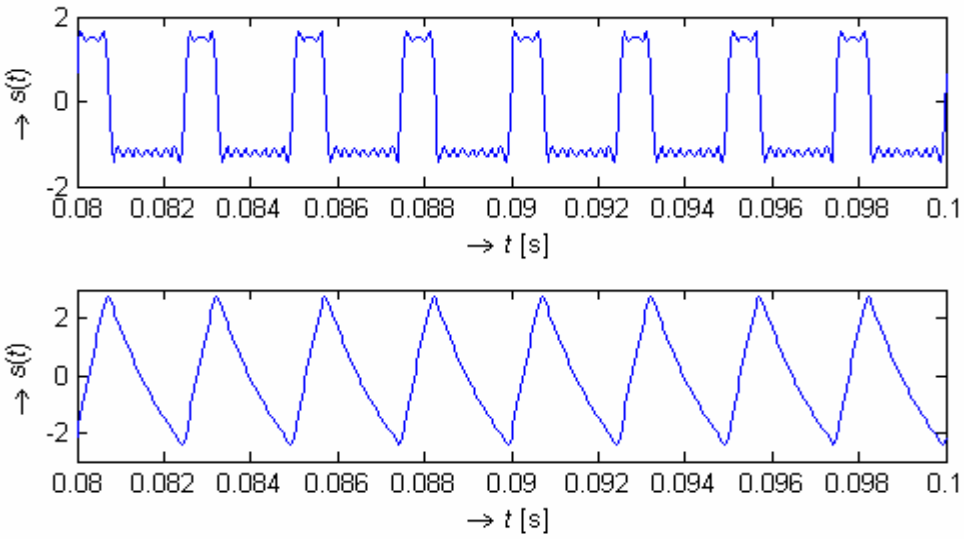
$$g(f,d) = \frac{2f}{d(1-d)}\,, \tag{2.17}$$

kde f je kmitočet v jednotkách počet cyklů za vzorek a d je střída v intervalu od 0 do 1. Na obr. 2.9 je znázorněn získaný trojúhelníkový signál. Nastavíme-li střídu obdélníkového průběhu na polovinu periody, získáme trojúhelníkový signál, který má stejnou strmost nástupné i sestupné hrany. Změníme-li střídu obdélníkového signálu, můžeme získávat proměnlivou strmost nástupné a sestupné hrany.

Teoreticky je tímto způsobem možné získávat pilový průběh. Uvedený způsob se ovšem příliš nevyužívá, protože je nutné použít dvě integrace a při druhé integraci může nastat problém se stejnosměrnou složkou signálu [1].



Obr. 2.8: Princip generování obdélníkového a trojúhelníkového průběhu.



Obr. 2.9: Ukázka vygenerovaného trojúhelníkového signálu získaného z obdélníkového průběhu s 10 harmonickými složkami.

2.5.4 Problémy se stejnosměrnou složkou

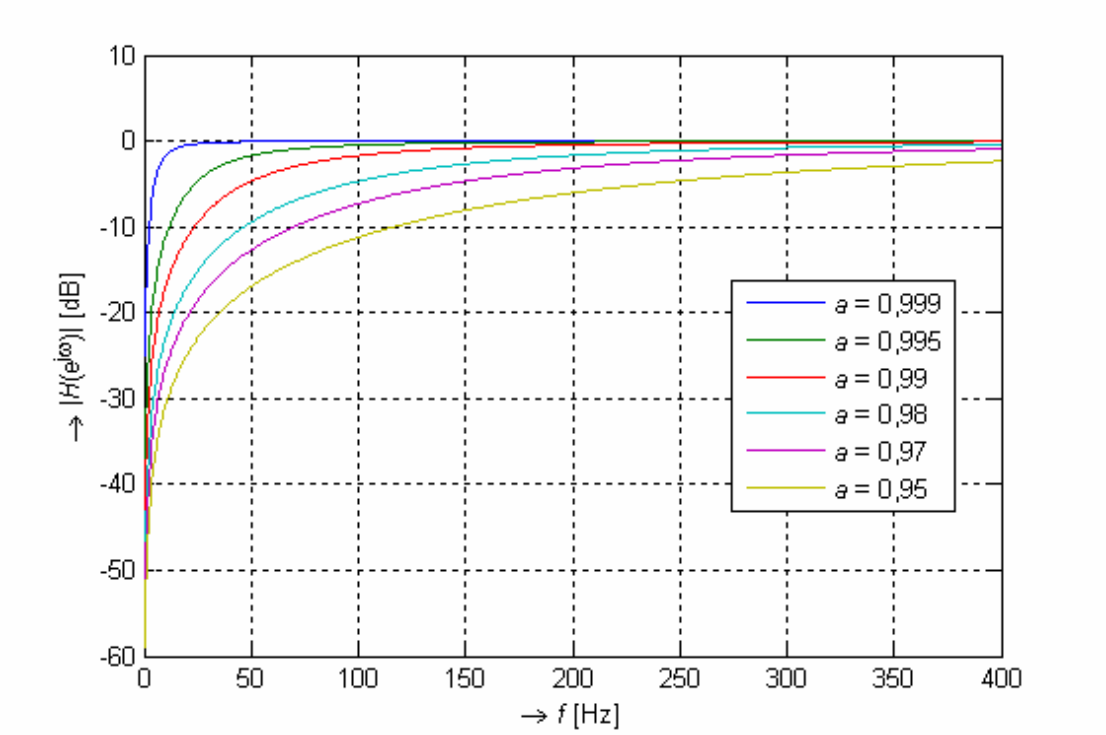
Aby nedošlo k saturaci na výstupu integrátoru, musí být vstupní signál integrátoru bez stejnosměrné složky. Jak již bylo zmíněno, metoda BP-BLIT nemá žádnou stejnosměrnou složku, takže není třeba žádné kompenzace pro výstupní obdélníkový průběh. Počáteční podmínky prvního integrátoru však mohou produkovat určitou stejnosměrnou složku na výstupu integrátoru, ta musí být odstraněna před druhou integrací.

Hodnota této stejnosměrné složky je závislá na střídě signálu. Při druhé integraci je zase závislá amplituda trojúhelníkového průběhu na kmitočtu, její velikost musí být korigována během změny kmitočtu. Je důležité si uvědomit, že stav integrace těsně před změnou se chová jako počáteční podmínky integrátoru po změně parametrů. Jednou

z možností, jak odstranit stejnosměrnou složku mezi první a druhou integrací, je vložení filtru, který odstraňuje stejnosměrnou složku (DC blocking filter) s přenosovou funkcí [3]

$$H(z) = \frac{z-1}{z-a}. \tag{2.18}$$

Parametr a se nastavuje blízký jedničce, aby docházelo k blokování pouze stejnosměrné složky, jak ukazuje obr. 2.10. Pro číslicové zpracování zvuku se většinou nastavuje parametr a na 0,995, protože pokles o 3 dB je přibližně na kmitočtu 30 Hz.



Obr. 2.10: Modulové kmitočtové charakteristiky filtru blokujícího stejnosměrnou složku pro různé hodnoty parametru a.

„Leaky“ integrátory

Při použití přesného integrátoru může představovat problém přítomnost numerické chyby způsobené například zaokrouhlováním. Tyto chyby se pak hromadí v integrátoru, což způsobuje nežádoucí kompenzaci signálu, která může vést k neschopnosti vytvářet požadovaný signál. Nejvíce jsou tyto chyby patrné při druhé integraci [1]. Z tohoto důvodu se využívají tzv. „leaky“ integrátory (nepřesné integrátory). Jsou vytvořeny tak, že posuneme póly digitálního filtru, který realizuje integraci, z jednotkové kružnice mírně dovnitř do jednotkové kružnice. Tyto integrátory mají exponenciální impulzovou odezvu, která postupně konverguje k nule. V nich dochází k postupnému „zapomínání“ špatné výchozí počáteční podmínky a numerických chyb.

2.6 Metody generování pásmově omezeného sledu impulzů

Některé z dříve uvedených metod lze použít ke generování pásmově omezeného sledu impulzů (BLIT – Band Limited Impulse Train), ale samozřejmě mohou být tyto metody použity i pro generování jiných průběhů signálu.

2.6.1 Součet funkcí sinc a sinc_M

Základní operací před vzorkováním je průchod signálu přes anti-aliasingový filtr. Ideální anti-aliasingový filtr má spojitou impulzovou odezvu [1]:

$$h_{vz} = \text{sinc}\left(f_{vz}t\right) = \frac{\sin\left(\pi f_{vz}t\right)}{\pi f_{vz}t}. \tag{2.19}$$

Ideální jednotkový sled impulzů s periodou T_1 je dán vztahem

$$x(t) = \sum_{l=-\infty}^{\infty} \delta\left(t + lT_1\right), \tag{2.20}$$

použitím anti-alisangového filtru s impulzovou odezvou h_{vz} na tento signál dostaneme

$$x_f(t) = \left(x * h_{vz}\right)(t) = \sum_{l=-\infty}^{\infty} \text{sinc}\left(t + lT_1\right), \tag{2.21}$$

kde $P = T_1/T_{\text{vz}}$, což je perioda ve vzorcích. Nyní je signál x_f pásmově omezen a může být vzorkován bez aliasingového zkreslení

$$y[n] = x_f(nT) = \sum_{l=-\infty}^{\infty} \text{sinc}(n + lP). \tag{2.22}$$

Výše uvedený výraz pro $y[n]$ může být někdy vykládán jako tzv. *časový aliasing funkce sinc* o délku P vzorků. Nekonečná suma je převedena do uzavřené formy

$$y[n] = \frac{M}{P}\,\text{sinc}_M\left(\frac{M}{P}n\right), \tag{2.23}$$

kde

$$\text{sinc}_M(x) = \frac{\sin(\pi x)}{M\sin(\pi x/M)}. \tag{2.24}$$

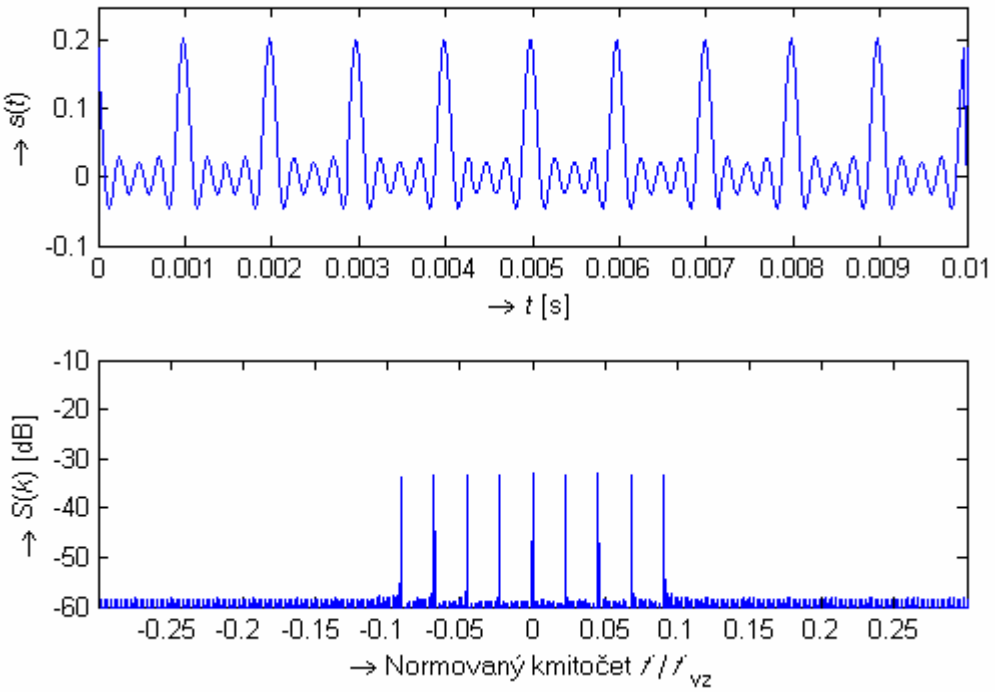
Po dosazení vztahu (2.24) do vztahu (2.23) dostaneme

$$y[n] = \frac{\sin(\pi nM/P)}{P\sin(\pi n/P)}. \tag{2.25}$$

Tato funkce může být využita pro přímou syntézu pásmově omezeného sledu impulzů (BLIT) podobným způsobem jako DSF. Proměnná P je perioda ve vzorcích,

M je počet spektrálních složek v oboustranném spektru. Do hodnoty M se započítává i stejnosměrná složka, proto musí být hodnota M vždy lichá, jelikož se jedná o reálný signál, který má sudý počet nenulových harmonických v dvoustranném spektru. Příklad vygenerovaného BLIT a jeho spektrum je znázorněno na obr. 2.11.

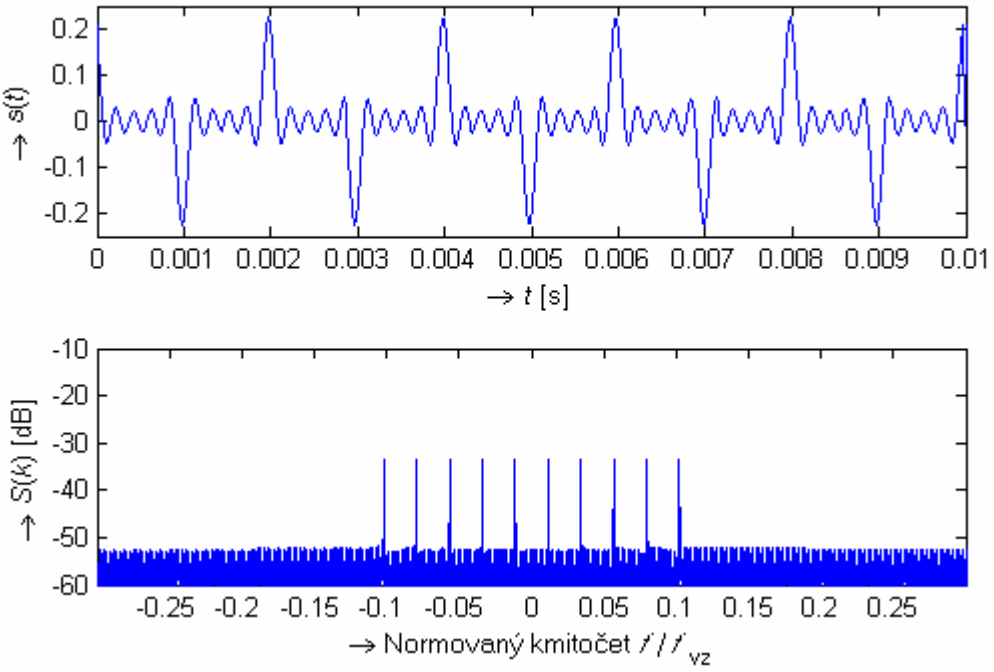
Nyní za předpokladu maximálního počtu harmonických (tj. do poloviny vzorkovacího kmitočtu) je poměr M/P vždy blízký 1. Když je P liché celé číslo a $P = M$, pak $y[n]$ je prostě $\mathrm{sinc}_M(n)$. Když se parametr P odchýlí od M, je potřeba kompenzovat amplitudu [9]. Tato metoda je velmi podobná metodě DSF v tom, že jde o uzavřenou formu s přesnou kontrolou počtu harmonických složek. Metoda DSF je častěji používána s menším počtem harmonických složek, které nezasahují do celého použitelného pásma, tedy do kmitočtu $f_{vz}/2$. V běžném provozu zůstává hodnota M stejná a přelaďuje se pouze kmitočet, tedy parametr P. V případě, že změníme hodnotu parametru M z jednoho celého čísla na jiné, ze vzorku na vzorek, může nastat nespojitost ve výstupním signálu, která je slyšitelná [1].



Obr. 2.11: Průběh BLIT s parametry $f_{vz} = 44100$, $M = 9$ a $P = 44{,}1$ a jeho spektrum.

K vytváření obdélníkového průběhu je potřeba bipolární sled pásmově omezených impulzů (BP-BLIT), jak bylo ukázáno dříve. Další možnou metodou, jak vytvořit bipolární sled pásmově omezených impulzů, je odečíst od sebe dva průběhy BLIT a jeden z nich zpozdit o požadovanou fázi. Další metoda je založena na změně parametru M

z lichého čísla na sudé, díky tomu je odstraněna stejnosměrná složka a dochází k přímé generaci BP-BLIT jak je zobrazeno na obr. 2.12. Vygenerovaný signál má poloviční kmitočet proti původnímu BLIT signálu a střídu 50 %.



Obr. 2.12: Průběh BP-BLIT s parametry $f_{vz} = 44100$, $M = 10$ a $P = 44{,}1$ a jeho spektrum.

2.6.2 Metoda diskrétního součtového vzorce

Jelikož metoda DSF má volnější definici, můžeme její pomocí generovat BP-BLIT mnohem flexibilněji [1], [6]. Za prvé musíme zaměnit sinus v definici rovnice DSF za kosinus (v podstatě nahrazení sinu za kosinus v čitateli, jmenovatel zůstává stejný). BLIT lze tedy generovat podle rovnice

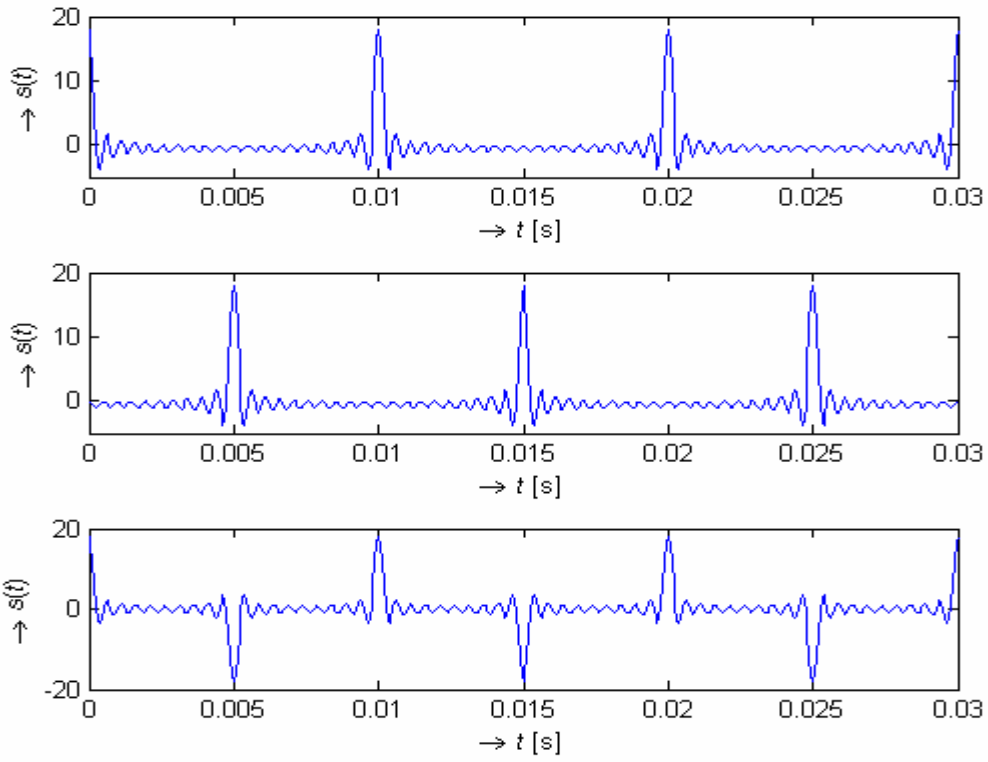
$$y(t) = \sum_{k=1}^{Num} a^k \cos(0 + 2\pi f \cdot kt), \tag{2.26}$$

kde N udává počet harmonických.

První krok, který je třeba pro vytvoření bipolárního pásmově omezeného sledu impulzů, je, že parametr a v definičním vztahu pro DSF bude záporný. Tím dojde k posunutí průběhu proti průběhu s kladným parametrem a přesně o polovinu periody. Poté již stačí tyto dva průběhy odečíst a dostaneme BP-BLIT se střídou 50 %, jak ukazuje obr. 2.13 a následující rovnice

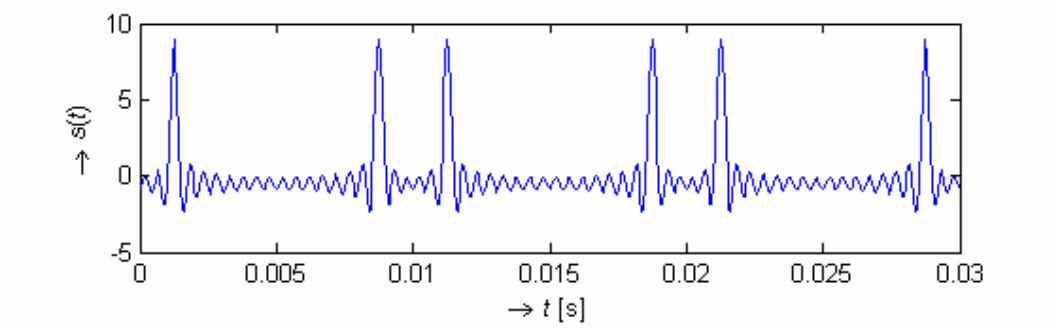
$$y(t) = \sum_{k=1}^{Num} a^k \cos(0 + 2\pi f \cdot kt) - \sum_{k=1}^{Num} (-a)^k \cos(0 + 2\pi f \cdot kt). \tag{2.27}$$

Pro jinou střídu než je 50%, je zajímavá metoda získání BP-BLIT. Parametr a se stane komplexním číslem ($a \cdot e^{j\varphi}$). Bipolární BLIT je získán jako imaginární část průběhu, jak to ukazuje obr. 2.14. Tato metoda je v podstatě stejně výpočetně náročná jako metoda odečtení dvou reálných průběhů BLIT získaných DSF s určitou fází, ale je matematicky „elegantnější“. V některých systémech, kde není implementována komplexní aritmetika, však může být jednoduší použít pouhé odečtení dvou reálných průběhů BLIT.

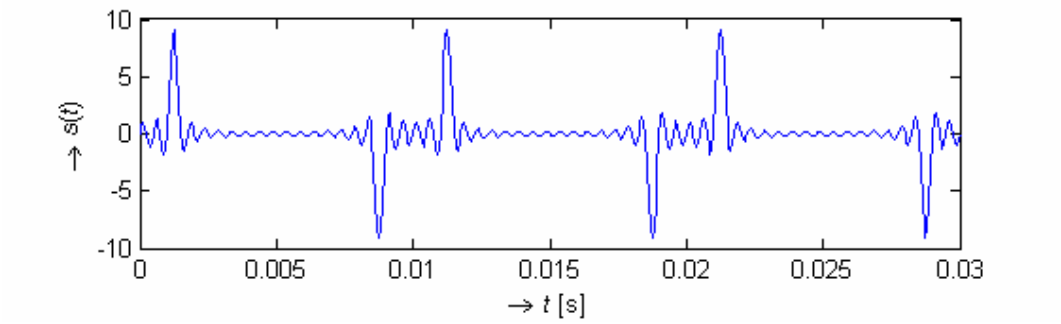


Obr. 2.13: Od shora: průběh BLIT s kladným parametrem a, průběh BLIT se záporným parametrem a, jejich výsledný rozdíl pro $N = 20$.

$$y(t) = \mathrm{Re}\left[\sum_{k=1}^{Num}\left(0{,}99e^{j\frac{\pi}{4}}\right)^{k}\cos(0+2\pi f\cdot kt)\right]$$



$$y(t) = \mathrm{Im}\left[\sum_{k=1}^{Num}\left(0{,}99e^{j\frac{\pi}{4}}\right)^{k}\sin(0+2\pi f\cdot kt)\right]$$



Obr. 2.14: Použití komplexního parametru a pro generaci BP-BLIT pro $N = 20$.

2.6.3 Součet funkcí sinc váhovaných oknem

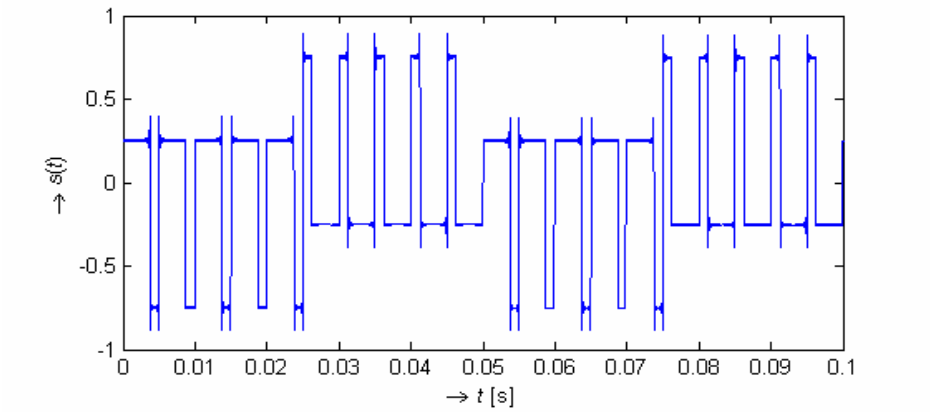
Další efektivní metodou pro syntézu pásmově omezeného sledu impulzů je metoda BLIT-SWS (Sum of Windowed Sinc). Tato technika je založena na tom, že každý jednotkový impulz $\delta(n)$ je nahrazen funkcí sinc váhovanou oknem $w(t)h_{vz}(t)$. Každý impulz je odděleně generován a přidán do výstupního signálu. Když je perioda generovaného signálu kratší než délka funkce sinc váhované okénkovou funkcí, budou se funkce sinc překrývat. Výběr okénkové funkce má znatelný význam na kvalitu generovaného pásmově omezeného sledu impulzů. Metoda je ve své podstatě ekvivalentní s pásmově omezenou tabulkovou metodou, protože funkce sinc váhovaná okénkovou funkcí je uložena do tabulky a z ní vyčítána do výstupu systému [5], [9], [10].

3 POROVNÁNÍ METOD

Tato kapitola se věnuje porovnání jednotlivých popsaných metod a to především z hlediska změn parametrů v reálném čase. Nejvýznamnější jsou impulzní šířková modulace nebo přeladťování z jednoho kmitočtu na jiný pomocí lineární změny kmitočtu. Dále jsou porovnány metody z hlediska výpočetní a paměťové náročnosti.

3.1 Změna parametrů v reálném čase

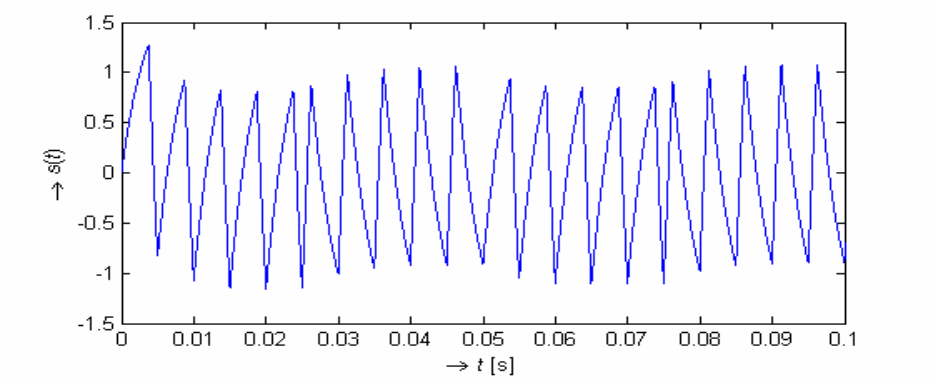
Jedním takovým parametrem, který je nutné měnit v reálném čase, je střída obdélníkového signálu, například při impulzní šířkové modulaci, což je velmi obtížně realizovatelné při použití metody aditivní syntézy a tabulkové metody. Tato metoda totiž také při vytváření složitějších signálů využívá aditivní syntézu. Na obr. 3.1 je zobrazen obdélníkový průběh vygenerovaný metodou integrace BP-BLIT, u které byl pásmově omezený bipolární sled impulzů byl generován metodou DSF.



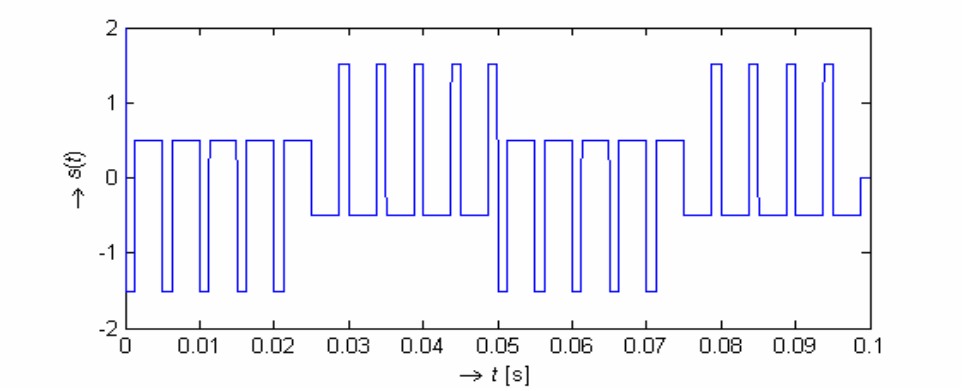
Obr. 3.1: Obdélníkový průběh se skokovou změnou střídy, vygenerovaný metodou integrace BP-BLIT vytvořený pomocí DSF s f_{vz} = 44,1 kHz.

V uvedeném průběhu dochází ke skokové změně střídy obdélníkového signálu podle modulačního obdélníkového průběhu (PWM – Pulse Width Modulation). Vygenerovaný obdélníkový signál nemá stejnosměrnou složku, aby bylo možné jeho integrací získat nesymetrický trojúhelníkový signál, který ukazuje obr. 3.2. K integracím byly využity tzv. „leaky“ integrátory, o kterých bylo pojednáno v kapitole 2.5.4. Jsou to v podstatě IIR filtry s exponenciální impulzovou odezvou, které postupně „zapomínají“ na špatné počáteční podmínky integrace. Na obr. 3.3 je zobrazen obdélníkový signál se změnou střídy v poměru modulačního obdélníkového signálu, který je vygenerován metodou DPW. Metody založené na derivaci po částech parabolických průběhů mají minimální

stejnosměrnou složku, jak bylo uvedeno v kapitole 2.4.2. Nevýhodou metody DPW je nemožnost měnit sklon nástupné a sestupné hrany nesymetrického trojúhelníkového průběhu.

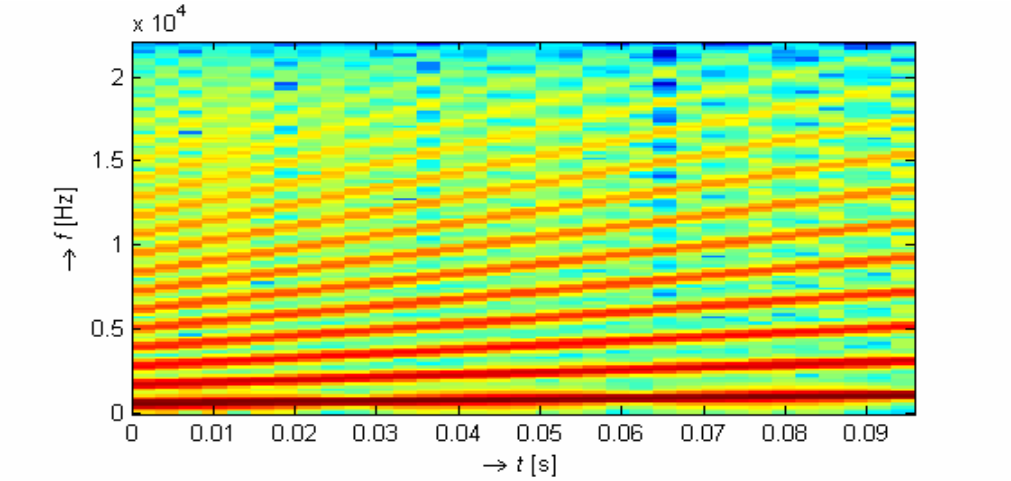


Obr. 3.2: Nesymetrický trojúhelníkový průběh se změnou sklonu hran, získaný integrací obdélníkového průběhu zobrazeném na obr. 3.1 s f_{vz} = 44,1 kHz.

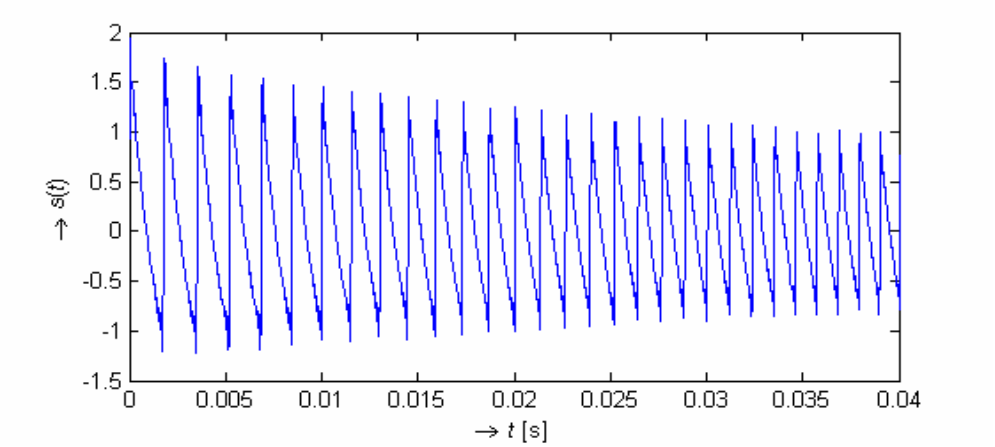


Obr. 3.3: Obdélníkový průběh se skokovou změnou střídy, vygenerovaný metodou DPW s f_{vz} = 44,1 kHz.

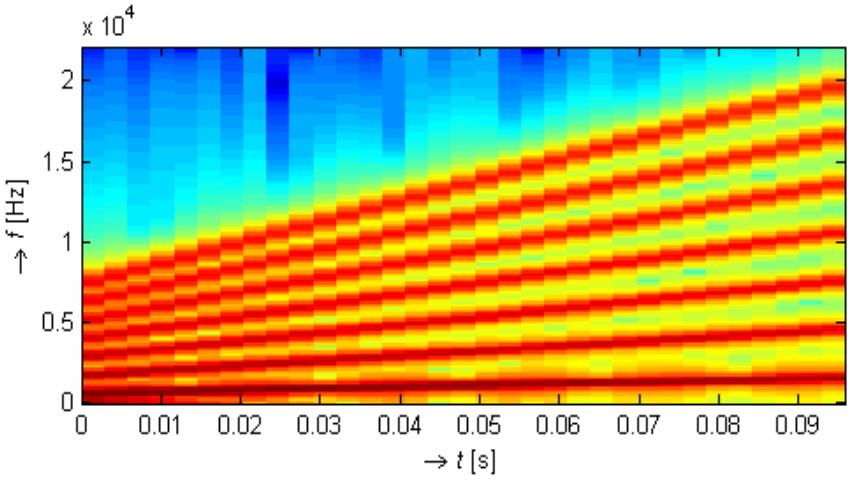
V hudebních syntezátorech je funkce zvaná „portamento“, což je plynulá změna základního kmitočtu generovaného tónu z jedné hodnoty na druhou. Proto byly jednotlivé metody porovnány také z hlediska tohoto parametru. Na obr. 3.4 je znázorněn spektrogram lineární změny kmitočtu obdélníkového průběhu se střídou 50%, průběh je vygenerován metodou DPW, jak spektrogram dokládá, signál má určité minimální aliasingové zkreslení. U metody postupné integrace BLITu dochází lineárním přelaďováním k exponenciálnímu zmenšování amplitudy, jak dokládá obr. 3.5, což je způsobeno použitím zmiňovaných „leaky“ integrátorů. Lineární změnu kmitočtu u metody BLIT ukazuje spektrogram na obr. 3.6. U této metody není využito celé použitelné pásmo, které zasahuje až do poloviny vzorkovacího kmitočtu. Dalo by se využít v případě, že bychom v každém okamžiku počítali a měnili přesný počet možných spektrálních složek.



Obr. 3.4: Lineární změna kmitočtu obdélníkového průběhu se střídou 50%, získaného metodou DPW.



Obr. 3.5: Lineární změna kmitočtu pilového signálu, vygenerovaného integrací BLIT s f_{vz} = 44,1 kHz.



Obr. 3.6: Lineární změna kmitočtu obdélníkového průběhu se střídou 50%, získaného metodou integrace BP-BLIT s f_{vz} = 44,1 kHz.

3.2 Teoreticky zhodnocená výpočetní náročnost

Výpočetní náročnost je důležitá z důvodu implementace některého algoritmu v mobilních audio procesorech a dalších levných aplikacích, které mají omezené výpočetní a paměťové prostředky. V tab. 3.1 je uvedeno neformální porovnání některých metod syntézy pásmově omezených signálů seřazených podle výpočetní náročnosti [4].

Výpočetně nejméně náročným algoritmem je metoda syntézy DPW, která ve svém základu využívá jednoduchý modulo čítač, který má malé nároky na výpočetní prostředky. Dále už metoda využívá pouze jednoduché operace a FIR filtr druhého řádu, naopak má tato metoda celkem nízkou zvukovou kvalitu (viz dále).

Výpočetně velmi náročnou metodou je generace pomocí pásmově omezeného sledu impulzů, který je generován metodou DSF. To se projeví zejména při generování signálů s nízkým základním kmitočtem, protože je nutné sčítat velké množství harmonických složek. Tabulková metoda syntézy má ze všech metod největší paměťovou náročnost, protože pro generování kvalitního zvukového signálu je potřeba mít v tabulce dostatečné množství vzorků jedné periody.

Tab. 3.1: Porovnání jednotlivých metod syntézy z hlediska výpočetní a paměťové náročnosti a z hlediska zvukové kvality.

Metoda	**Výpočetní náročnost**	**Paměťová náročnost**	**Zvuková kvalita**
Aditivní syntéza	Velmi vysoká	Velmi malá	Vynikající
BLIT-DSF	Velmi vysoká	Velmi malá	Vynikající
BLIT-SWS	Průměrná až vysoká	Malá	Velmi dobrá
Tabulková syntéza	Průměrná až vysoká	Velmi velká	Velmi dobrá
DPW	Velmi malá	Velmi malá	Dobrá

4 IMPLEMENTACE METOD PRO GENEROVÁNÍ SIGNÁLU V REÁLNÉM ČASE

Tato kapitola bude věnována implementaci jednotlivých metod syntézy do VST plug-in modulu (Virtual Studio Technology). Technologie VST plug-in modulů využívá jazyk C++, proto zde bude probrána implementace jednotlivých metod v tomto programovacím jazyku. První podkapitola se zabývá obecným popisem technologie VST. Další část kapitoly již řeší implementaci metod do VST plug-in modulu. K vývoji VST plug-in modulu byla využita šablona VST plug-in modulu vytvořená na Ústavu telekomunikací FEKT VUT v Brně s doplněnými některými funkcemi.

4.1 Technologie VST

Technologie VST využívá tzv. zásuvné moduly (plug-in) pro číslicové generování a zpracování zvukových signálů. Byla vyvinuta německou firmou Steinberg a poprvé představena v roce 1996. Tato technologie a jí podobné používané v digitálním zpracováním zvuku mají softwarově simulovat tradiční nahrávací studia. Protože o rok později se z ní stal otevřený standard, byly vyvinuty stovky volně dostupných, ale i komerčních plug-in modulů [11]. VST plug-in modul není aplikace, ale pouze softwarový modul, který se připojuje k aplikaci. Ta pro plug-in modul spravuje vstupní a výstupní data a je nazývána *hostitelskou aplikací*. Tato aplikace předá modulu tok dat, ten jej určitým způsobem modifikuje nebo generuje nový tok dat a předá jej zpět hostitelské aplikaci. Ta nemá žádné informace o algoritmu plug-in modulu a ani je nepotřebuje. Tok dat je rozdělen na segmenty, které jsou jednotlivě předávány plug-in modulu prostřednictvím volání funkce rozhraní VST. Délku segmentu určuje hostitelská aplikace [12], [13].

4.1.1 Integrované vývojové prostředí

VST SDK (Virtual Studio Technology Software Development Kit) je sada vývojových nástrojů, vyvinutá pro vývojáře VST plug-in modulů, která by měla usnadnit jejich práci a je volně dostupná na internetových stránkách firmy Steinberg. Starší VST technologie 2.0 podporuje až osmikanálové plug-in moduly s na sobě nezávislým počtem vstupů a výstupů, která podporuje 32 bitový formát s pohyblivou řádovou čárkou v rozsahu od -1 do 1. Je však vhodnější stáhnout po novější verzi VST SDK 2.4, která již

umožňuje zpracovávat data v 64 bitovém rozlišení s plovoucí řádovou čárkou. Nejnovější verzí je verze 3.0, která je zcela přepracovaná a jejíž programový model je složitější [12].

V jakém vývojovém prostředí budeme VST SDK používat, je omezeno specifickými vlastnostmi platformy a zejména operačním systémem. Na operačním systému jsou závislé binární formáty knihoven plug-in modulu (dynamicky připojovaná knihovna ve Windows, tzv. raw Code ressource v MacOS 9, tzv. bunde v MacOSX a knihovna v BeOS a Unixu) a také grafické funkce používané knihovnou VST GUI. Zdrojové soubory VST SDK jsou v jazyce C++. Proto v operačním systému Windows můžeme použít například kompilátor MS Visual C++ nebo Borland Builder C++ a na platformě Mac OS X kompilátor prostředí XCode. Oficiálním vývojovým prostředím pro VST SDK 2.4 je prostředí MS Visual C++ 8 pro operační systém Windows [12].

4.1.2 Ovladače ASIO

Ovladač zvukové karty ASIO (Audio Stream Input/Output) je určen pro počítačové zpracování zvukových signálů. Ovladač byl vyvinut firmou Steinberg a vytváří rozhraní s velmi nízkým dopravním zpožděním mezi aplikací a počítačovou zvukovou kartou.

Výrobci zabývající se výrobou profesionálního hardwaru pro počítačové zpracování hudby standardně dodávají ovladače ASIO ke svým zařízením. Mezi tyto výrobce patří například firmy: MOTU, R.M.E., M-Audio, ESI, Edirol, Echo Audio, E-Mu, Behringer a další. Tito výrobci své zvukové karty dodávají pro nejrůznější rozhraní jako jsou PCI, PCIe, USB nebo IEEE1394.

Pro zvukové karty s integrovaným kodekem AC'97 lze využít ovladač Asio4all dostupný z [14]. Tento ovladač v podstatě zapouzdří ovladač WDM (Windows Driver Model) ve Windows a zvuková karta se pak tváří pro hostitelskou aplikaci, jako by měla ovladače ASIO [12].

4.1.3 Hostitelská aplikace

Hostitelská aplikace je jednoduchou aplikací umožňující načíst vytvořený VST plug-in modul. Volně dostupných hostitelských aplikací je celá řada jako například MiniHost [15] nebo Audacity [16]. O tom, kterou aplikaci použijeme, rozhoduje mnoho požadavků. Jedním z nich může být například schopnost spuštění aplikace v ladícím režimu, který ovšem podporuje jen málo hostitelských aplikací. Takovou hostitelskou aplikací je například GT Player EDU, který je součástí VST šablony a jeho vzhled je

zobrazen na obr. 4.1. Tato aplikace využívá pro vstup a výstup zvukových dat ovladače technologie ASIO a podporuje plug-in moduly technologie VST. Při spuštění aplikace v ladícím režimu vývojového prostředí (debug) dojde k přeložení knihovny (.dll), kde je umožněno krokování řádek po řádku. Tato knihovna není ovšem optimalizovaná, což nám při krokování nevadí. Naopak při zkompilování knihovny v konfiguraci „release“ je knihovna optimalizována a není umožněno krokování. Tento režim se využívá pro tvorbu finálního sestavení knihovny [12].

Obr. 4.1: Hostitelská aplikace GT Player EDU pro testováni VST plug-in modulů.

4.2 Implementace metod do VST šablony

Tato podkapitola popisuje implementací některých metod syntézy uvedených v teoretické části do VST šablony vytvořené na Ústavu telekomunikací FEKT VUT v Brně. První část je věnována obecnému nastavení VST šablony a následující části se zabývají jednotlivými funkcemi generujícími průběhy pomocí výše popsaných metod syntézy.

4.2.1 Nastavení v souborech šablony pro vytvoření požadovaného plug-in modulu

Nejdříve musíme v hlavičkovém souboru `vst_temp_defs.h` definovat makra a výčtové typy šablony, které definují základní vlastnosti plug-in modulu [17]. Jedním z těchto parametrů je výčtový typ `tagTemplateParametr`, který definuje pojmenované indexy parametrů zobrazené v GUI:

- `kGain = 0` – změna zesílení (počátek číslování vždy od nuly)
- `kFreg` – změna kmitočtu generovaného signálu
- `kDuty` – změna střídy generovaného signálu
- `kSquare` – změna typu signálu na obdélníkový průběh

- `kTriangle` – změna typu signálu na trojúhelníkový průběh
- `kBypass` – předefinovaná konstanta pro přemostění generátoru
- `kBLITSWS` – generování průběhů metodou BLIT-SWS
- `kBLITDSF` – generování průběhů metodou BLIT-DSF
- `kDPW` – generování průběhů metodou DPW

Dalším souborem určeným k editaci uživatele je hlavičkový soubor `user_variables.h`, ve kterém jsou deklarovány proměnné. V nich jsou uloženy hodnoty po skončení jednoho vygenerovaného segmentu a jsou určeny jako počáteční hodnoty pro další segment. Seznam těchto proměnných využitých v plug-in modulu generátoru je uveden v tab. 4.1. Téměř všechny proměnné jsou polem o velikosti definovaného počtu kanálů (`def_ChNumber`), protože je potřeba ukládat počáteční hodnoty pro segmenty zvlášť pro každý kanál.

Tab. 4.1: Seznam uživatelských proměnných deklarovaných v `user_variables.h`.

Proměnná	Účel proměnné
`initialfilter_in[def_ChNumber]`	Počáteční podmínky filtrace – vstupní vzorky
`initialfilter_out[def_ChNumber]`	Počáteční podmínky filtrace – vyfiltrované vzorky
`buffer_store[def_ChNumber]`	Uložení aktuálních hodnot vyrovnávací paměti u metody BLIT-SWS
`bufferposition[def_ChNumber]`	Pozice při vyčítání z vyrovnávací paměti u metody BLIT-SWS
`position[def_ChNumber]`	Čítač vyčtených vzorků do výstupního signálu u metody BLIT-SWS
`initialgeneration[def_ChNumber]`	Počátek generace signálu následujícího segmentu
`sinc_table`	Tabulka vzorků funkce sinc váhované okénkovou funkcí

Editace souboru `vst_temp.cpp`

V tomto souboru je vymezena část určená k editaci uživatele komentáři `USER CODE` a `END OF USER CODE` [17]. Zde nejdříve vkládáme hlavičkové soubory potřebné pro námi vytvořený kód, definujeme zde také konstanty plug-in modulu, jako jsou například název a identifikátor modulu. Do pole struktur `unsortedParams[PARAM_NUMBER]` definujeme vlastnosti a chování nastavovacích prvků, které budou zobrazeny v GUI. Parametry, které jsme definovali ve výčtovém typu `tagTemplateParametr`, budou nyní odpovídat vždy jednomu prvku pole struktur. Pole struktur bude definováno takto:

```
// tag, aktuální, výchzí, min, max, název, popis, typ, pouze ke čtení
{kGain, 0.0f, 0.5f, 0.0f, 1.0f, "Gain", "dB", SHOW_IN_DB, false},
{kFreq, 0.0f, 1000.0f, 30.0f, 20050.0f, "Freq", "Hz", SHOW_LOG, false},
{kDuty, 0.0f, 0.5f, 0.0f, 1.0f, "Duty", "", SHOW_FLOAT, false},
{kSquare, 0.0f, 0.0f, 0.0f, 1.0f, "Square", "", SHOW_ONOFF, false},
{kTriangle, 0.0f, 0.0f, 0.0f, 1.0f, "Triang", "", SHOW_ONOFF, false},
{kBypass, 0.0f, 0.0f, 0.0f, 1.0f, "Bypass", "", SHOW_ONOFF, false},
{kBLITSWS, 0.0f, 0.0f, 0.0f, 1.0f, "BlitSWS", "", SHOW_ONOFF, false},
{kBLITDSF, 0.0f, 0.0f, 0.0f, 1.0f, "BlitDSF", "", SHOW_ONOFF, false},
{kDPW, 0.0f, 0.0f, 0.0f, 1.0f, "DPW", "", SHOW_ONOFF, false},
```

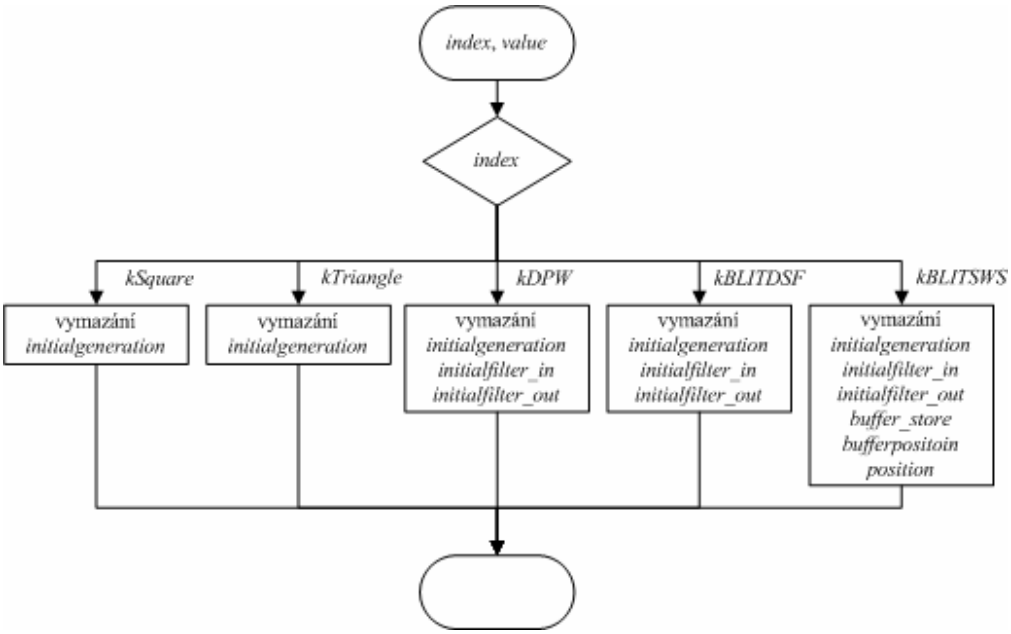
Popis bude viditelný ve vytvořeném GUI a může mít maximálně 8 znaků. Typ definuje zobrazení parametru [17]:

- `SHOW_FLOAT` – zobrazení datového typu s plovoucí řádovou čárkou
- `SHOW_LOG` – zobrazení datového typu float v logaritmické stupnici
- `SHOW_IN_DB` – číslo datového typu float je přepočteno na dB
- `SHOW_ONOFF` – dvoupolohový vypínač (zapnuto/vypnuto)

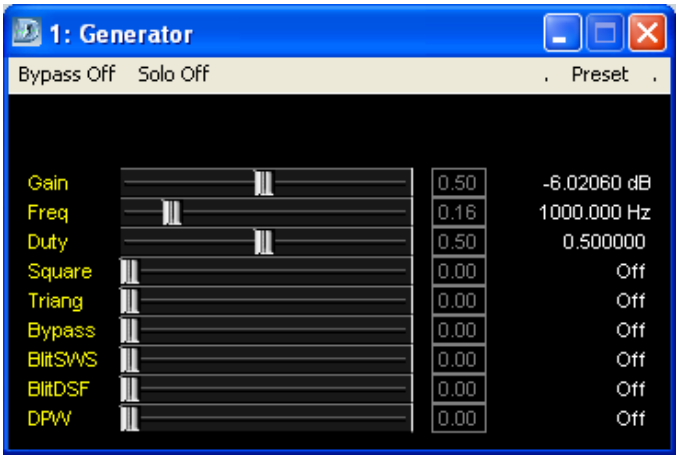
V metodě `setUserVariables` dochází k dynamické alokaci paměti proměnných definovaných v hlavičkovém souboru `user_variables.h`. Navíc je zde volána funkce pro vyplnění sinc tabulky `lookuptable`, která bude vysvětlena v kapitole 4.2.4. Metoda `clearUserVariables` slouží k dealokaci paměti dynamických uživatelských proměnných. Metoda `parameterChanged` je volána vždy při změně parametru, pokud vrátí `true` je změna povolena, jestliže `false` zakázána. Tato metoda je často využívána při implementaci parametrického filtru, kde po změně vstupních parametrů je třeba vypočítat nové koeficienty filtru. V případě implemetace generátoru je využita k vynulování počátečních podmínek filtrace a generace, aby nedošlo například ke změně střídy obdelníkového průběhu při přepínání jednotlivých druhů průběhů. Na obr. 4.2 je zobrazen vývojový diagram této metody. Nejdůležitější metodou v souboru `vst_temp.cpp` je metoda `processReplacing`, která má tři vstupní parametry. Prvními dvěma jsou ukazatele na ukazatele vstupního a výstupního pole vzorků (`inputs`, `outputs`) a posledním třetím je délka zpracovávaného segmentu dat (`sampleFrames`).

V metodě jsou nejprve získány nastavené parametry (`kFreq`, `kGain`, `kDuty`) generovaného signálu pomocí metody `getRealValue` a uloženy do proměnných `f`, `G` a `duty` a jsou předány jednotlivým funkcím, ve kterých jsou implementovány jednotlivé metody syntézy, stejně jako získaný vzorkovací kmitočet dodaný metodou `getSampleRate` uložený v proměnné `fs`. Podle nastavené metody generace je volaná příslušná funkce definovaná v hlavičkovém souboru `generator.h`. Metodám je také předáván parametr `type`, který definuje nastavený typ průběhu (pilový, obdélníkový nebo

trojúhelníkový). Při zapnutí metody syntézy posuvníkem je generován ve výchozím nastavení pilový průběh a obdélníkový nebo trojúhelníkový průběh je spouštěn posuvníky „Sguare“ a „Triang“. Po zkompilování projektu je vytvořeno editorem GUI, podle nastavených vlastností a chování v poli struktur `unsortedParams`, ukázka vygenerovaného GUI je zobrazena na obr. 4.3.



Obr. 4.2: Vývojový diagram metody `parameterChanged`.



Obr. 4.3: Vygenerované GUI generátoru.

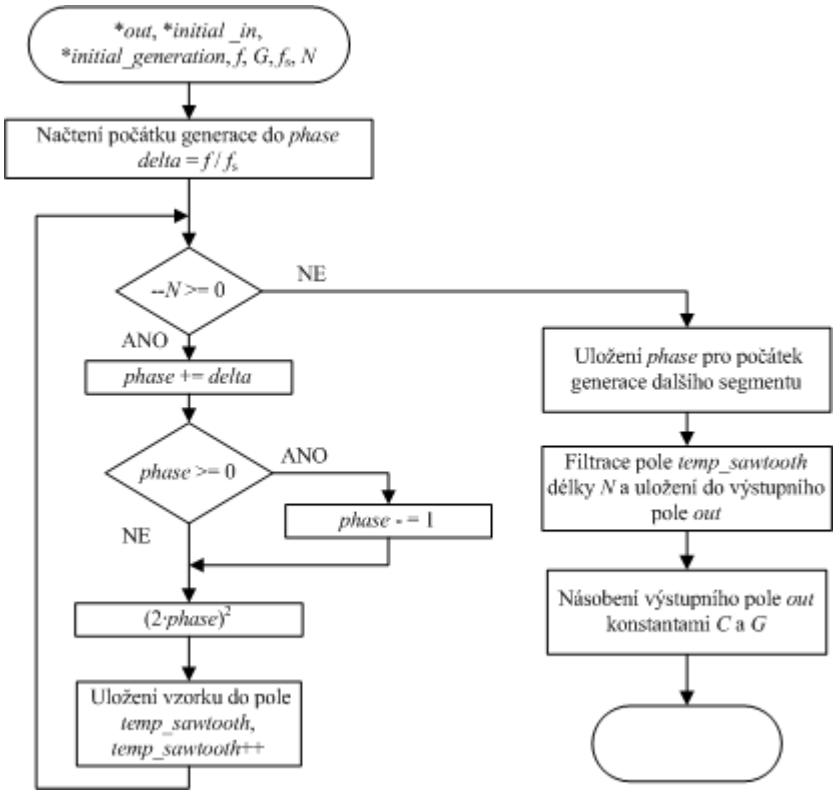
4.2.2 Implementace metody derivace parabolického průběhu

Metoda syntézy derivace po částech parabolického průběhu, která byla popsána v kapitole 2.4, je definována funkcí v hlavičkovém souboru `generator.h` takto:

`void dpw(float *out, float *initial_generation, int *tr_square, float *initial_in, float*initial_in2, float f, float G, float fs, float duty, VstInt32 N, SignalType type_signal).`

- `out` – ukazatel na pole výstupních vzorků
- `initial_generation` – počáteční hodnota, od které se má začít generovat průběh
- `initial_in` – uložení počátečních podmínek filtrace pro pilový průběh
- `initial_in2` – uložení počátečních podmínek filtrace pro druhý pilový průběh sloužící k vytvoření obdélníkového průběhu
- `duty` – střída obdélníkového průběhu
- `N` – délka segmentu
- `type_signal` – typ generovaného signálu (`SAWTOOTH, SQUARE, TRIANGLE`)

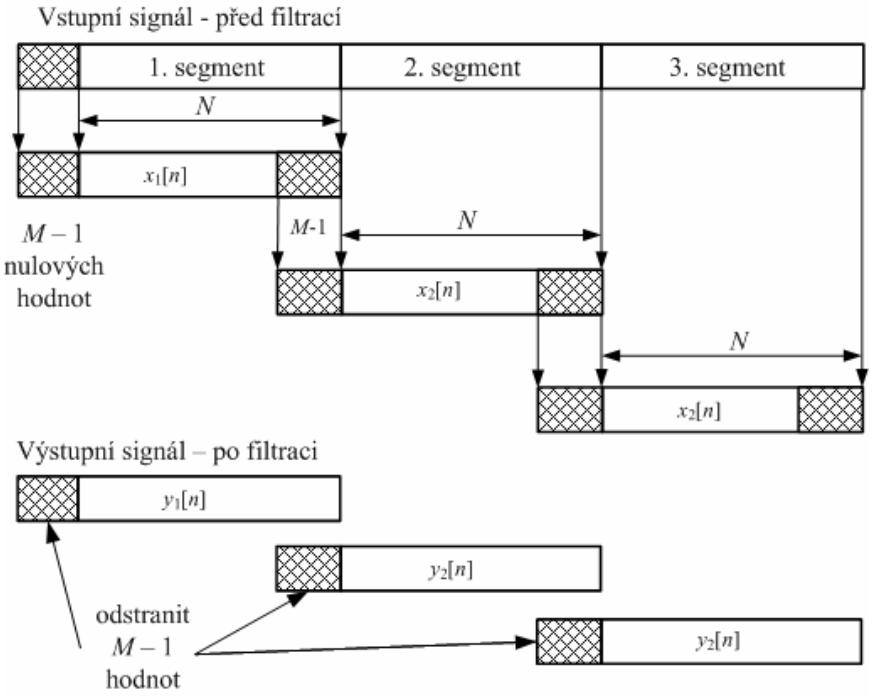
V této funkci jsou volány jednotlivé funkce pro generování požadovaného průběhu, volba se provádí proměnnou `type_signal`. Algoritmus generování pilového průběhu je patrný z vývojového diagram zobrazeného na obr. 4.4.



Obr. 4.4: Algoritmus generace pilové průběhu v metodě `generate_triangle_dpw`.

Obdélníkový průběh je získán odečtením dvou pilových průběhů s nastavenou střídou a trojúhelníkový průběh je získán mírnou modifikací tohoto algoritmu uvedenou v kapitole 2.4.3. U všech použitých metod je potřeba po vyfiltrování celého segmentu

uložit počáteční hodnoty filtrace pro další segment, což jsou poslední vzorky předešlého segmentu. Tento princip se nazývá metodou odstranění přesahu [18] a její podstata je naznačena na obr. 4.5, kde M je délka impulzové charakteristiky $h[n]$ a N je délka segmentu. Platí, že N je daleko větší než M. V případě metody DPW je délka impulzové charakteristiky M FIR filtru rovna třem. Tedy jak je ukázáno na obr. 4.5, stačí ukládat M-1 hodnot pro další segment, proto v našem případě stačí pouze dvě hodnoty [18].



Obr. 4.5: Podstata metody odstranění přesahu [18].

4.2.3 Implementace metody pásmově omezeného sledu impulzů vytvořeného pomocí diskrétního součtového vzorce

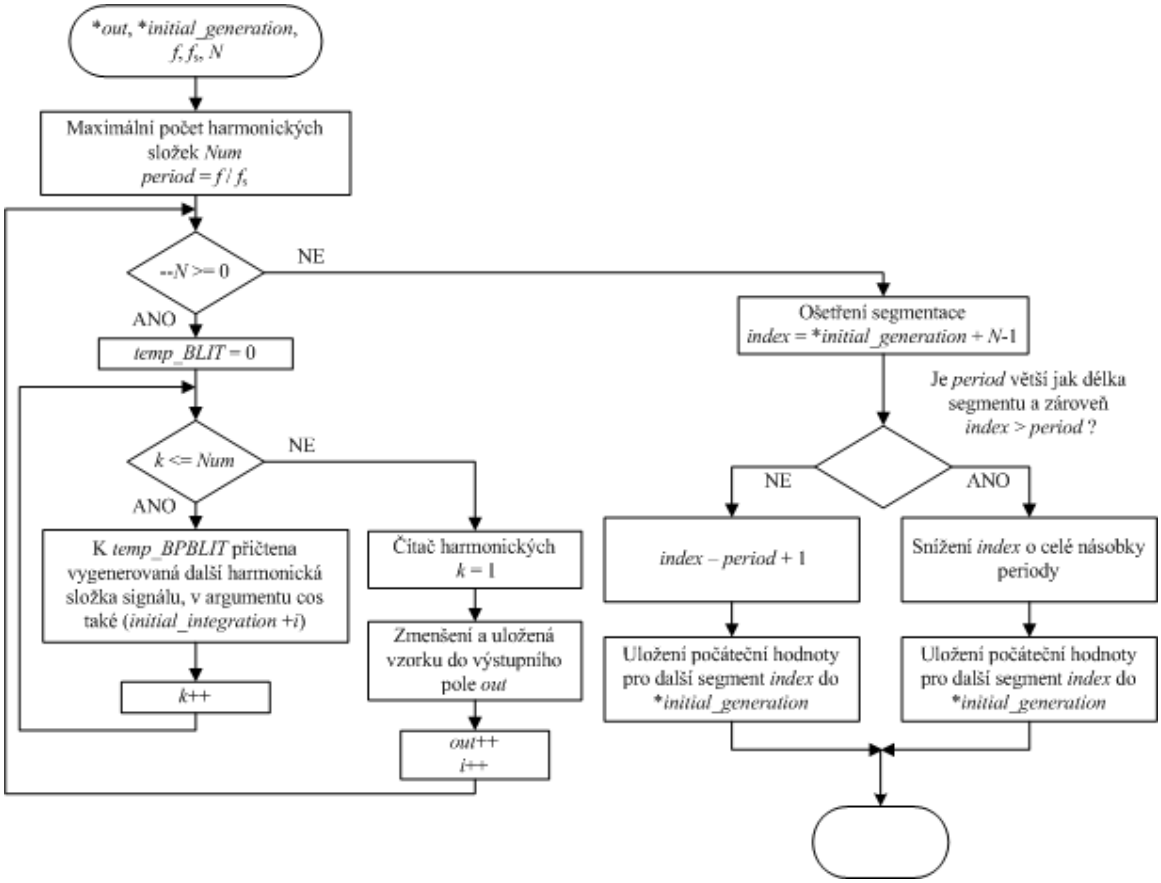
Funkce v níž je implementována metoda postupných integrací pásmově omezeného sledu impulzů je opět uložena v souboru `generator.cpp` a je definována v hlavičkovém souboru `generator.h` takto:

```
void blit_dsf(float *out, float *initial_generation, float *initial_out, float *initial_in, float f, float G, float fs, float duty, VstInt32 N, SignalType type_signal).
```

Význam jednotlivých vstupních proměnných je stejný jako u předcházející metody a opět dochází k přepínání druhu signálu podle proměnné `type_signal`. Při generování obdélníkového signálu je třeba vytvořit průběh bipolárního pásmově omezeného sledu impulzů, jeho vytváření je patrné z vývojového diagramu zobrazeného na obr. 4.6. Poté již stačí BP-BLIT integrovat a získáme obdélníkový průběh s požadovanou střídou.

Způsoby generování dalších druhů signálů, jak pilového, tak trojúhelníkového byly popsány v kapitole 2.5. Při generování je nutné zajistit, aby proměnná v argumentu funkce sinus `initial_generation` v podstatě zastupující čas, nenarůstala do nekonečna, proto je zde vytvořeno ošetření snižující tuto proměnnou, které je patrné také z vývojového diagramu znázorněného na obr. 4.6. Rovnice pro generaci BP-BLIT byla upravena, aby nebylo nutné využívat komplexní čísla pomocí Moivreovy věty [19] následovně:

$$y(t) = \mathrm{Im}\left[\sum_{k=1}^{Num}\left(0{,}99e^{j\frac{\pi}{4}}\right)^{k}\sin\left(0+2\pi f\cdot kt\right)\right] = \sum_{k=1}^{Num}0{,}99^{k}\cdot\sin\left(k\frac{\pi}{4}\right)\cdot\sin\left(0+2\pi f\cdot kt\right). \quad (4.1)$$



Obr. 4.6: Algoritmus generace BP- BLIT v metodě `generate_bpblit`.

4.2.4 Implementace metody pásmově omezeného sledu impulzů vytvořeného pomocí součtu okénkovaných funkcí sinc

U této metody je pásmově omezený sled impulzů vytvořený součtem okénkovaných funkcí sinc. Funkce je definována v hlavičkovém souboru takto:

```
void blit_sws(float *out, float *buffer_store, int *buffer_pos,
Cycle *position, float *initial_out, float *initial_in, float f, float
G, float fs, float duty_cycle, VstInt32 N, SignalType type_signal,
float *sinc_table).
```

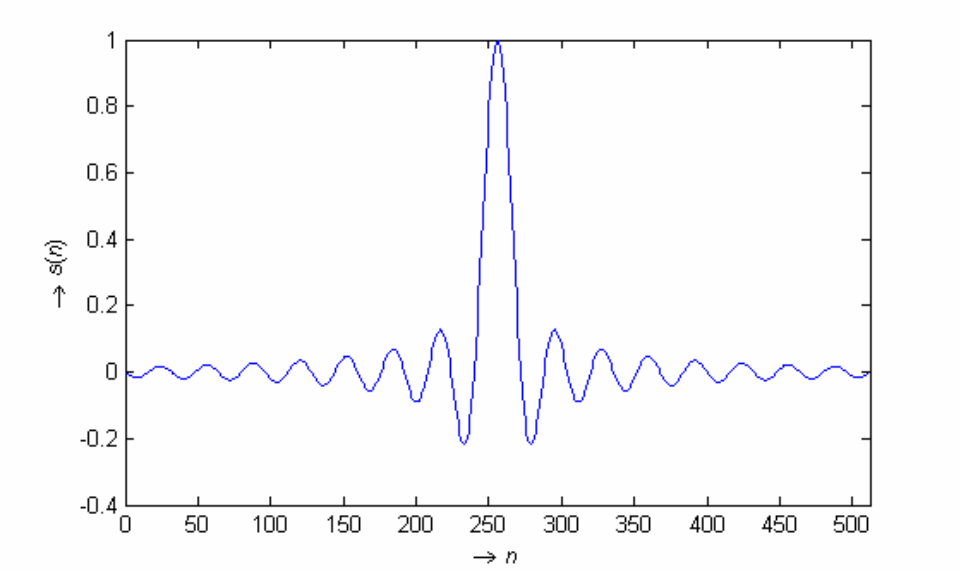
Pole `sinc_table` obsahuje tabulku s uloženými hodnotami funkce sinc váhované vhodným okénkem. K vyplnění tohoto pole je při inicializaci proměnných volána funkce `lookuptable`, která vytvoří funkci sinc podle zadaných parametrů a omezí ji Blackmanovým oknem, které je definováno vztahem [20]:

$$w[n] = a_0 - a_1 \cos\left(\frac{2\pi n}{N-1}\right) + a_2 \cos\left(\frac{4\pi n}{N-1}\right), \tag{4.2}$$

kde $a_0 = 0{,}42$, $a_1 = 0{,}5$ a $a_2 = 0{,}08$. Parametry, které slouží ke generaci funkce sinc, jsou definovány v souboru `generator.h` takto:

- `SINC_NBR_ZEROS` – udává počet průchodů nulou funkce sinc
- `SINC_RESOL` – udává počet bitů, na kolik má být jeden úsek mezi průchody nulou funkce sinc kódován (např pro 8 bitů je 256 vzorků na jeden úsek mezi průchody nulou funkce sinc)

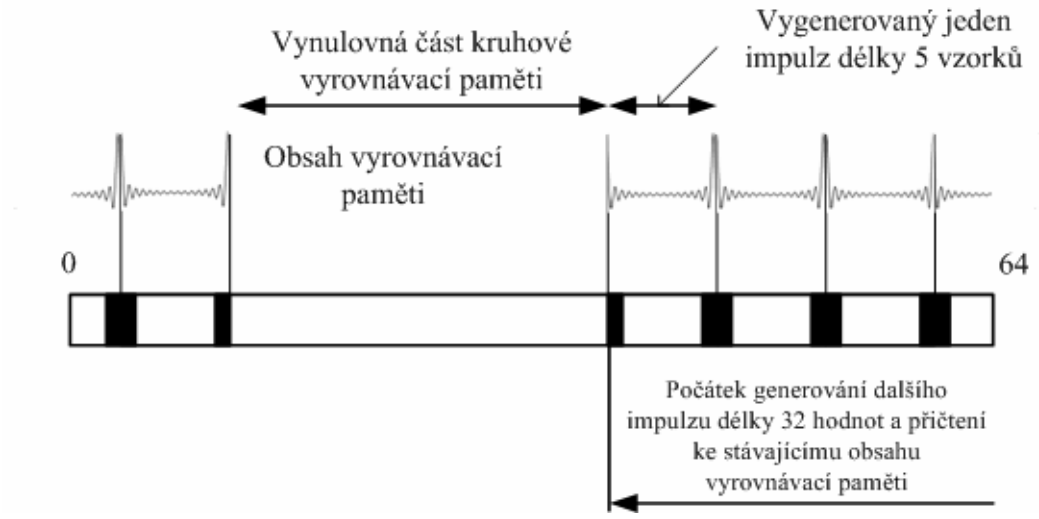
Těmito parametry a okénkovou funkcí lze ovlivnit kvalita generovaného signálu. Obvykle má jeden úsek mezi dvěma průchody nulou funkce sinc 128 až 1024 vzorků [1].



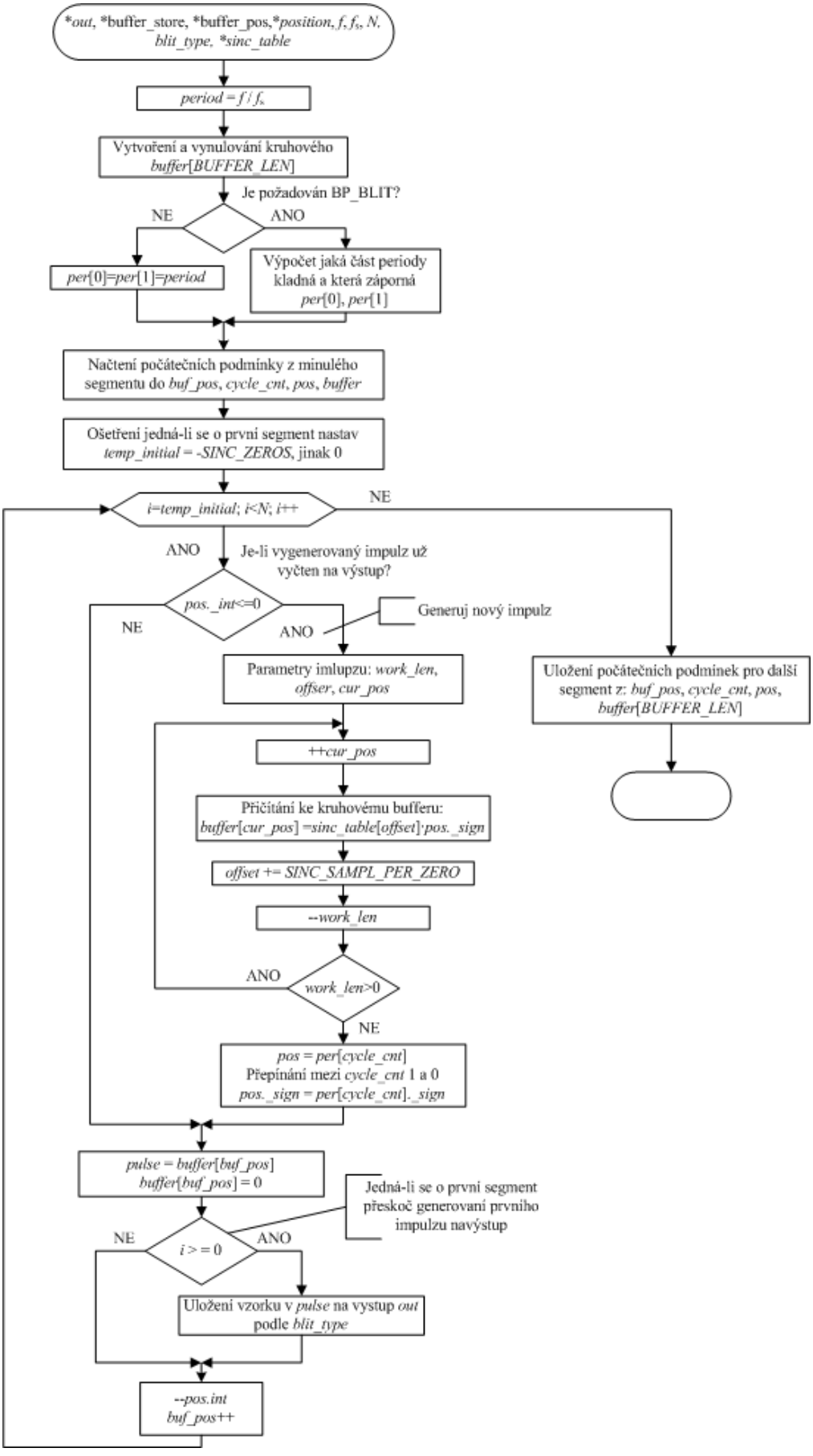
Obr. 4.7: Hodnoty funkce sinc s 512 vzorky omezené Blackmanovým. oknem uložené v poli `sinc_table`.

Ukázka hodnot pole `sinc_table` je na obr. 4.7, ten zobrazuje funkci sinc s 32 průchody nulou, kde každý úsek mezi průchody nulou má 16 vzorků, což dohromady dává tabulku s 512 vzorky funkce sinc omezené Blackmanovým oknem. Ve funkci

`blit_sws` opět dochází k přepínání do funkcí pro generaci podle nastavené proměnné `type_signal`. Při vytváření průběhu pásmově omezeného sledu impulzů podle vývojového diagramu na obr. 4.9, je využita kruhová vyrovnávací paměť reprezentovaná polem `buffer[BUFFER_LEN]`. Jestliže je perioda generovaného průběhu menší než uvedená vyrovnávací paměť, jsou postupně přičítány k obsahu vyrovnávací paměti další generované impulzy, jak zobrazuje obr. 4.8. Pro uložení délky periody a čítač výstupních vzorků je definována struktura `Cycle` v hlavičkovém souboru `generator.h`. Tato struktura obsahuje tři položky. První `_int` pro uložení celočíselné velikosti periody, druhou `_frac` pro uložení desetinné části velikosti periody a položku `_sign`, která je využívána při generování bipolárního pásmově omezeného sledu impulzů k určení, ve které části periody se nacházíme, zda v kladné nebo záporné. Ostatní potřebné průběhy jsou ze standardního a bipolárního pásmově omezeného sledu impulzů opět získávány způsobem popsaným v kapitole 2.5.



Obr. 4.8: Princip generování a přičítání pulzů ke kruhové vyrovnávací paměti.



Obr. 4.9: Algoritmus generace BP-BLIT a BLIT v metodě `generate_blit_sws`.

5 POROVNÁNÍ METOD IMPLEMENTOVANÝCH V TECHNOLOGII VST

Závěrečná kapitola bude věnována porovnání jednotlivých metod syntézy implementovaných do VST plug-in modulu. V první podkapitole budou metody porovnány z hlediska celkového harmonického zkreslení a v následující kapitole z hlediska výpočetní náročnosti.

5.1 Celkové harmonického zkreslení

Metody byly porovnány z hlediska celkového harmonického zkreslení, tj. změřením napětí harmonických složek náležejících do spektra měřeného průběhu a aliasingových složek a následným vypočtením zkreslení podle rovnice (5.1). Sestava měření se skládala ze spektrálního analyzátoru Hewlett-Packard HP 35665A a z počítače, na kterém byl spuštěný program GT Player EDU, který umožňoval načtení vytvořeného VST plug-in modulu.

$$\mathrm{THD} = \frac{\sum U_{\text{Aliasingových složek}}^{2}}{\sum U_{\text{Harmrmonických složek náležejících do spektra}}^{2}} \cdot 100 \qquad (5.1)$$

Byla použita externí zvuková karta M-Audio MobilePre s USB rozhraním. Hladina šumu u této zvukové karty byla na zjištěna na spektrálním analyzátoru a měla hodnotu přibližně 20 μV. Tab. 5.1 shrnuje naměřené a vypočtené výsledky zkreslení pro jednotlivé implementované metody syntézy a pro průběhy obdélníkové a pilové.

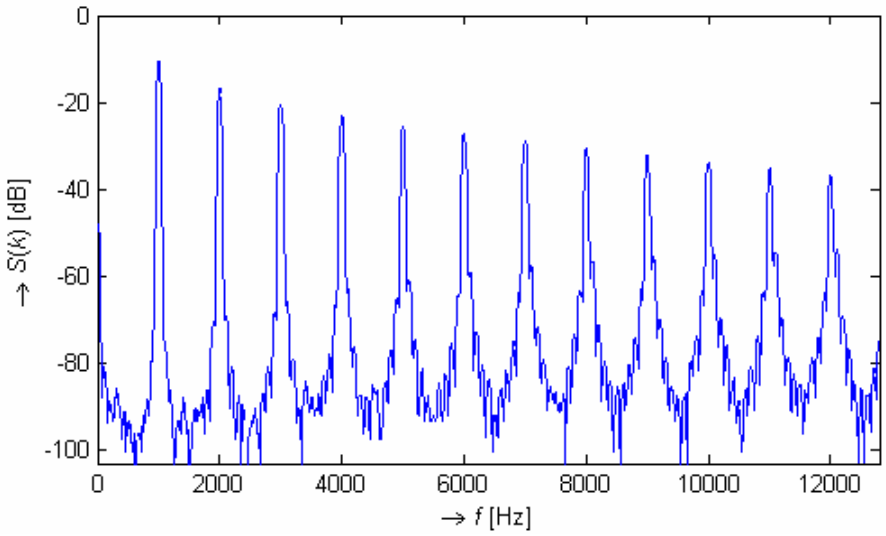
Tab. 5.1: Porovnání jednotlivých metod syntézy z hlediska zkreslení.

Metoda syntézy	**Zkreslení pilového průběhu [%] (střída 0,5)**	**Zkreslení obdélníkového průběhu [%]**
BLIT-DSF	0	0
BLIT-SWS	0,0012	0,0004
DPW	0,0405	0,0266

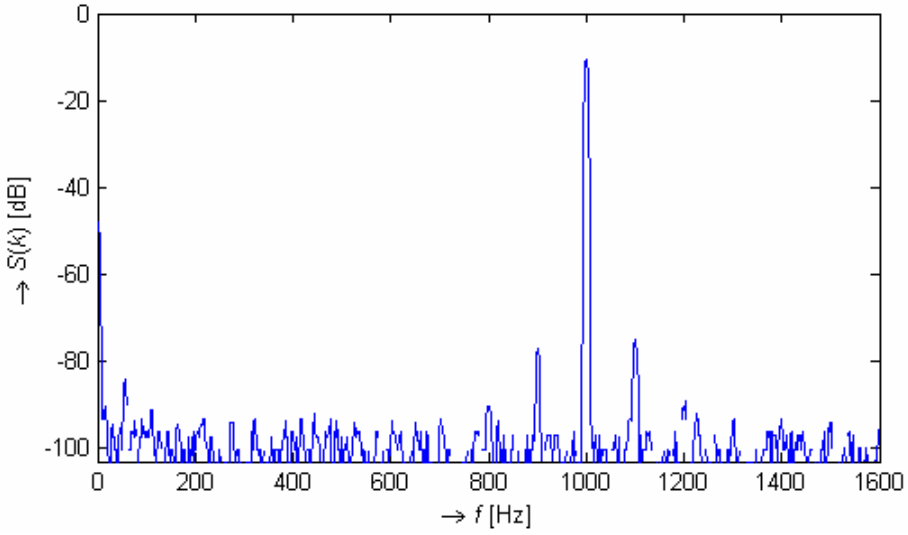
Trojúhelníkové průběhy nebyly u metod proměřovány, protože, jak bylo dříve uvedeno, mají nejprudší pokles velikostí spektrálních složek, a tedy mají i nejmenší aliasingové zkreslení. Z tabulky je patrné, že hodnota zkreslení u metody generace BLIT-SWS je o více než jeden řád lepší než syntéza metodou DPW. Metoda generace pásmově

omezeného sledu impulzů pomocí diskrétního součtového vzorce (BLIT-DSF) nevykazovala žádné aliasingové zkreslení.

Parametry metody součtu funkcí sinc váhovaných oknem (BLIT-SWS) byly: 128 průchodů nulou a 256 vzorků na interval mezi dvěma průchody nulou. Bylo použito Blackmanovo okno. Na obr. 5.1 je zobrazeno spektrum pilového průběhu generovaného metodou derivací po částech parabolického průběhu (DPW) o kmitočtu 1 kHz zasaženého aliasingovým zkreslením. Obr. 5.2 zobrazuje detailnější pohled na spektrum tohoto pilového průběhu, a to především na první harmonickou složku tohoto průběhu a dvě postranní aliasigové složky.



Obr. 5.1: Spektrum pilového průběhu o kmitočtu 1 kHz generovaného metodou DPW a změřeného spektrálním analyzátorem.



Obr. 5.2: Detailní pohled na první harmonickou složku pilového průběhu o kmitočtu 1 kHz generovaného metodou DPW a změřeného spektrálním analyzátorem.

5.2 Výpočetní náročnost

Všechna následující měření byla prováděna při vypnutém šetřícím režimu a na počítači v následující konfiguraci:

CPU: AMD Athlon 64 2800+
Taktovací kmitočet 1800 MHz
L1 Cache: 64 kB + 64 kB (instrukce + data)
L2 Cache: 512 kB

RAM: 1 GB DDR

OS: Microsoft Windows XP Professional Service Pack 3

Nejdříve byla měřeno zatížení procesoru (CPU – Central Processing Unit) jednotlivými metodami syntézy, výsledky uvádí tab. 5.2. Měření byla prováděna v programu GT Player. Nejdříve bylo změřeno zatížení procesoru metodami a od této hodnoty byla odečtena hodnota změřená při překlenutí generátoru funkcí bypass plug-in modulu. Takto bylo měření prováděno proto, aby byla vyloučena režie programu GT Player. Dále bylo zjištěno, že u metody postupných integrací pásmově omezeného sledu impulzů (BLIT-DSF) je možné generovat signál na daném testovacím počítači v reálném čase až od kmitočtu přibližně 350 Hz u pilového průběhu a zhruba od kmitočtu 480 Hz u obdélníkového a trojúhelníkového průběhu, což je způsobeno potřebným součtem velkého počtu harmonických složek u této metody.

Tab. 5.2: Naměřené hodnoty zatížení CPU metodami syntézy pro: signál o kmitočtu 1 kHz, délkou rámce 2048 vzorků a vzorkovacím kmitočtu 44,1 kHz.

Metoda	**Pilový průběh [%]**	**Obdélníkový průběh (střída 0,5) [%]**	**Trojúhelníkový průběh [%]**
BLIT-DSF	30	42	42
BLIT-SWS	1	2	2
DPW	1	1	1

Nejdůležitějším měřením pro zpracování signálu v reálném čase je měření poměrné doby potřebné k vygenerování určitého počtu vzorků. Byla měřena potřebná doba k vygenerování 2048 vzorků nastavenou metodou, byl generován signál o kmitočtu 1 kHz a různých průběhů. K měření byly využity hodnoty systémového času resp. hodnoty čítače s vysokým rozlišením (High-Resolution Performance Counter v MS Windows [21]). který dosahuje přesnosti až 1 μs. Všechna měření doby potřebné

k vygenerování 2048 vzorků byla prováděna v pěti opakováních, z těchto pěti hodnot byla vypočtena průměrná hodnota, směrodatná odchylka a relativní chyba měření. Naměřené hodnoty jsou uvedeny v tab. 5.3 pro vzorkovací kmitočet 44,1 kHz a tab. 5.4 pro vzorkovací kmitočet 22,05 kHz. Při snížení vzorkovacího kmitočtu klesly doby potřebné k vygenerování 2048 vzorků u metody BLIT-DSF na polovinu, což je způsobeno součtem poloviční počtu potřebných harmonických složek. U ostatních metod syntézy se při snížení vzorkovacího kmitočtu jen nepatrně zvýšila doba potřebná k vygenerování rámce. Z naměřených hodnot plyne vysoká výpočetní náročnost metody BLIT-DSF, kterou dokládá i tab. 5.5 s vypočtenými hodnotami průměrného počtu matematických operací potřebnými k vygenerování jednoho vzorku signálu o kmitočtu 1 kHz. Tento počet matematických operací v sobě ale nezahrnuje rozdílné doby pro vyhledávání hodnot funkce sinc u metody BLIT-SWS nebo doby potřebné k výpočtu goniometrických funkcí u metody BLIT-DSF.

Tab. 5.3: Doba potřebná k vygenerování 2048 vzorků signálu o kmitočtu 1 kHz se vzorkovacím kmitočtem 44,1 kHz.

Metoda	**Pilový průběh**		**Obdélníkový průběh (střída 0,5)**		**Trojúhelníkový průběh**	
	t [ms]	**Chyba [%]**	**t [ms]**	**Chyba [%]**	**t [ms]**	**Chyba [%]**
BLIT-DSF	13,4222	0,2935	20,2288	0,8181	20,1346	0,4061
BLIT-SWS	0,2667	1,5076	0,4135	2,6586	0,4495	2,0336
DPW	0,1165	1,0496	0,2209	0,5341	0,1448	0,6539

Tab. 5.4: Doba potřebná k vygenerování 2048 vzorků signálu o kmitočtu 1 kHz se vzorkovacím kmitočtem 22,05 kHz.

Metoda	**Pilový průběh**		**Obdélníkový průběh (střída 0,5)**		**Trojúhelníkový průběh**	
	t [ms]	**Chyba [%]**	**t [ms]**	**Chyba [%]**	**t [ms]**	**Chyba [%]**
BLIT-DSF	6,8632	1,1746	10,4400	0,6749	10,3383	1,0480
BLIT-SWS	0,3636	0,9835	0,5618	0,3717	0,6011	0,6443
DPW	0,1134	1,4688	0,2438	0,8495	0,1583	1,4156

Tab. 5.5: Průměrný počet matematických operací potřebných na vygenerování jednoho vzorků signálu o kmitočtu 1 kHz se vzorkovacím kmitočtem 44,1 kHz.

Metoda	**Pilový průběh**	**Obdélníkový průběh (střída 0,5)**	**Trojúhelníkový průběh**
BLIT-DSF	235	302	299
BLIT-SWS	16	22	23
DPW	11	26	14

ZÁVĚR

V teoretické části práce byly rozebrány a popsány vybrané metody pro syntézu pásmově omezených číslicových zvukových signálů a jak těmito metodami generovat pilový, obdélníkový a trojúhelníkový signál.

Vybrané metody syntézy byly naprogramovány v prostředí Matlab. Zdrojové kódy skriptů jsou uvedeny na přiloženém CD. Metody byly porovnány z hlediska změn parametrů v reálném čase, především změny střídy obdélníkového průběhu a lineární změny kmitočtu. Bylo zjištěno, že u některých metod, jako je například aditivní syntéza a tabulková metoda, je jen velmi obtížně realizovatelná změna střídy obdélníkového průběhu v reálném čase. Proto se tyto metody absolutně nehodí jako zdroj nosného signálu pro impulzní šířkovou modulaci. Naopak u metod postupné integrace BLITu a metody DPW je změna střídy obdélníkového průběhu v reálném čase velice jednoduchá.

Metoda postupné integrace BLITu má problémy se stejnosměrnou složkou při integracích, ty se řeší použitím filtru blokujícího stejnosměrnou složku a tzv. „leaky“ integrátory. Metoda derivací parabolického průběhu nemá možnost změn dob trvání nástupné a sestupné hrany u nesymetrického trojúhelníkového průběhu, jak je to možné u metody postupných integrací BLITu.

Metody byly teoreticky porovnány z hlediska výpočetní a paměťové náročnosti na základě simulací v prostředí Matlab. Na základě simulací byly vybrány metody syntézy po částech parabolického průběhu DPW a metody postupné integrace pásmově omezeného sledu impulzů BLIT-DSF a BLIT-SWS k implementaci do plug-in modulu technologie VST, který umožňuje generování signálů v reálném čase.

U implementovaných metod byla změřena a vypočtena výpočetní náročnost. Dále bylo u těchto metod změřeno celkové harmonické zkreslení. Metodou, která má nejmenší výpočetní nároky, je metoda syntézy DPW. Tato metoda ovšem nemá tak dobrou kvalitu syntetizovaného zvukového signálu. Naopak výpočetně nejnáročnější je metoda postupné integrace BLIT, kde pásmově omezený sled impulzů byl vytvořen pomocí metody DSF. U metody je prakticky nulové aliasingové zkreslení. Nevýhodou uvedené metody je nemožnost generování signálů s nízkým kmitočtem v reálném čase. Jako nejperspektivnější metodou se jeví metoda BLIT, kde pásmově omezený sled impulzů je získán součtem funkcí sinc váhovaných oknem (BLIT-SWS). Metoda má nízké celkové harmonické zkreslení a také nízké výpočetní nároky.

POUŽITÁ LITERATURA

[1] STILSON, T., S. *Efficieantly-Variable Non-Oversampled Algorithms in Virtual-Analog Music Synthesis - A Root-Locus Perspective.* Ph.D. Thesis. Stanford University, 2006. 433 s. Dostupné na internetu: http://ccrma.stanford.edu/~stilti/.

[2] ŠEBESTA, V.; SMÉKAL, Z. *Signály a soustavy*: skripta. FEKT VUT v Brně, 2006. 165 s.

[3] SERRA, J., P. *Design and Implementation of Polyphonic Subtractive Synthesizer Using VST Plug-In Technology*: Diplomová práce. FEKT VUT v Brně, 2008. 76 s.

[4] VÄLIMÄKI, V.; HUOVILAINEN, A. Antialiasing Oscillators in Subtractive Synthesis. *IEEE Signal Processing Magazine*, Březen 2007, vol. 24, no. 2, s. 116-125. ISSN 1053-5888.

[5] SMITH, J.,O.; GOSSTT, P. A Flexible Sampling-Rate Conversion Method. In *Acoustic, Speech, and Signal Processing, IEEE International Conference on ICASSP '84*, Březen 1984, vol. 9, s. 112-115.

[6] MOORER, J., A. The synthesis of complex audio spectra by means of discrete summation formulas. *Journal of the Audio Engineering Society*, Prosinec 1975, vol. 24, s. 717-727.

[7] VÄLIMÄKI, V. Discrete-Time Synthesis of the Sawtooth Waveform With Reduced Aliasing, *IEEE Signal Processing Letters*, 2005, vol. 12, no. 3, s. 214–217. ISSN 1070-9908.

[8] VÄLIMÄKI, V.; HUOVILAINEN, A. Oscilator and Filter Algorithms for Virtual Analog Synthesis. *Computer Music Journal*, 2006, vol. 30, no. 2, s. 19-31.

[9] STILSON, T.; SMITH, J. Alias-Free Digital Synthesis of Classic Analog Waveforms. In *Proceedings of the 1996 International Computer Music Conference, Hong Kong*, Computer Music Association, 1996. 12 s. Dostupné na internetu: http://ccrma.stanford.edu/~stilti/.

[10] BRANDT, E. Hard Sync Without Aliasing. *In Proc. 2001 International Computer Music Conference*, 2001, s. 365-368. Dostupné na internetu: http://www.cs.cmu.edu/~eli/papers/

[11] STEINBERG. *The Steinberg Story* [online]. c2010 [cit. 2010-04-26]. Dostupné na internetu: <http://www.steinberg.net/en/company/steinberg_company_history.html>.

[12] SCHIMEL, J. *Implementace algoritmu číslicového zpracovaní signálů pomocí technologie VST: učební text.* FEKT VUT v Brně, 29. 4. 2009, s. 6.

[13] SCHIMEL, J. *Systém VST: učební text pro výuku předmětu MCSI.* FEKT VUT v Brně, 2009, s. 4.

[14] TIPPACH, M. *ASIO4ALL – Universal ASIO Driver For WDM Audio* [online]. aktualizováno 23. 10. 2008 [cit. 2010-04-26]. Dostupné na internetu: <http://www.asio4all.com/>

[15] FLEISCHER,T. *MiniHost 1.64* [online]. c2010 [cit. 2010-05-15]. Dostupné na internetu: <http://www.tobybear.de/p_minihost.html>

[16] MAZZONI, D.;DANNENBERG, R. aj. *Audacity 1.3* [online]. c2010 [cit. 2010-05-15]. Dostupné na internetu: <http://audacity.sourceforge.net/download/>

[17] SCHIMEL, J. *Šablona VST plug-in modulu: učební text*. FEKT VUT v Brně, 4. 5. 2008, dokument verze 1.2, s. 4.

[18] SMÉKAL, Z. *Číslicové zpracování signálů*: skripta. FEKT VUT v Brně, 6. 4. 2009, s. 203

[19] ABRAMOWITZ; MILTON & STEGUN; IRENE, A. *Handbook of Mathematical Functions.* New York: Dower Publications, 1964. 1020 s. ISBN 0486612724

[20] WEISSSTEIN E, W. *Concise encyclopedia of mathematics*. 2. vyd. Boca Raton: Chapman & Hall/CRC, 1998. 3205 s. ISBN 1-58488-347-2.

[21] *QueryPerformanceCounter Function* [online]. c2010 [cit. 2010-05-15]. Dostupné na internetu: <http://msdn.microsoft.com/en-us/library/ms644904%28v=VS.85%29.aspx>

SEZNAM PŘÍLOH

PŘÍLOHA

Příloha A: Obsah přiloženého CD

- **Diplomová práce** – text diplomové práce ve formátu pdf
- **Generátor** – adresář obsahuje soubory šablony VST a zdrojové kódy generátoru:
    - **editor** – v adresáři jsou umístěny soubory editor.h a editor.cpp se zdrojovými kódy editoru, který vytváří GUI podle zadaných parametrů
    - **source** – adresář obsahuje soubory:
        - *generátor.cpp* a *generátor.h* – zdrojové kódy generátoru
        - *vst_tem.cpp*, *vst_temp.h*, *vst_temp_defs.h* a *user_variables.h* – upravené zdrojové kódy šablony VST
    - **vst2.x** – adresář obsahuje zdrojové kódy potřebné pro vytvoření VST plug-in modulu. Dále obsahuje soubory:
    - **vstgui** – adresář obsahuje zdrojové kódy využité pro tvorbu GUI
    - **win** –adresář obsahuje soubory projektu vytvořené Visual Studiem. V podadresáři Release je výsledný zkompilovaný plug-in modul ve formě knihovny (*vst_temp_vc8.dll*). Dále je zde program VST Pluton Analyse pro testování plug in modulu a hostitelská aplikace GT Player EDU pro provoz VST plug-in modulu. V podadresáři Plugins jsou uloženy plug-in pro zobrazení průběhu a spektra signálů.
- **Matlab** – adresář se soubory s jednoduchými skripty potřebnými pro porovnání jednotlivých metod syntézy:
    - *AS_method.m* a *AS_retune_frequence* – metoda aditivní syntézy
    - *DPW_method*, *DPW_PWM* a *DPW_retune frequence* – metoda derivací po částech parabolického průběhu
    - *DSF_BLIT_method*, *DSF_BLIT_PWM* a *DSF_BLIT_retune_frequence* – metoda postupných integrací pásmově omezeného sledu impulzů